Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Networked Systems
ETH Zürich — seit 2015

NANOG Mailing List Analysis

Semester Thesis

Author: Lina Gehri

Tutors: Rüdiger Birkner, Alexander Dietmüller

Supervisor: Prof. Dr. Laurent Vanbever

March 2020 to June 2021

Abstract

Since 1992 network operators have been using the NANOG ("North American Network Operators' Group") mailing list to discuss best practices, get advice, complain about internet providers, and socialize with other network operators. The topics of the discussions range from routing problems over new technologies to provider recommendations. As such, the mailing list is an excellent indicator for former and current research questions, challenges, and technical advancements. However, no one has exploited this data yet, even though it could give many intriguing insights. This thesis presents how I apply natural language processing techniques to develop a topic model and conduct a sentiment analysis for all emails sent. I then use the topic and sentiment analysis as tools to dive into the mailing list, understand what mattered to network operators over the years, and determine how they felt about different topics.

Contents

Chapter 1

Introduction

Since the beginning of the Internet, there were network operators who had to configure it. Over the years, their tasks changed, new challenges emerged, and the topics that interested them evolved. One means of communication that over 9,500 networking operators have used since 1992 is the NANOG ("North American Network Operators' Group") mailing list. To this day, they have sent over 89,000 emails to this mailing list with topics ranging from security matters over hardware recommendations to routing table reports. As such, the mailing list is an indicator for former and current research questions, challenges, and technical advancements. However, no one has exploited this data yet, even though it could give many intriguing insights. All emails sent to the mailing list are openly accessible in an online archive. As it is impossible to read through 89,000 emails to get an overview of what network operators were concerned with, one has to use natural language processing techniques to process such an amount of text data.

This semester thesis aims to do this and discover what moved networking operators over the years. I want to find out what can be learned from the mailing list and if I can determine interesting trends with the help of natural language processing techniques. The final goal is to use the emails to answer interesting questions, like "Were network operators' sentiments about different providers affected by outages?" More specifically, the objective is to preprocess the emails and then create a topic model and conduct a sentiment analysis to use the two to explore the dataset.

The first step is to obtain the emails and prepare them in a suitable format. Because there are parts in a typical email that do not add value to an analysis and might even skew it, like signatures, confidentiality notices, or "answered to" sections, it is essential to clean the documents first. Therefore, before I start applying any natural language processing techniques, I remove these parts using regex patterns.

To understand what topics were important to network operators over time, I create a topic model. My approach is to use a document embedding model to map each email to a numeric vector and then use a clustering algorithm to identify topics. The challenge is to find the approach that works best for my use case. There are a few methods to build a topic model that assigns topics to documents, and I have to make sure that the model can deal with networking-specific vocabulary.

Another aspect of the emails I am interested in is the sentiment they convey. I want to learn how the authors felt about different topics during different periods. To add a numeric sentiment score to each email, I employ a library that uses a rule-based approach to calculate sentiment scores. Here, I have to consider the technical jargon of network operators again and adapt the vocabulary of the library accordingly.

The remainder of this report is structured as follows: Chapter 2 gives background information about the mailing list and summarizes the basic concepts of a natural language processing workflow.

In chapter 3, I show how I obtain and pre-process the emails before starting to analyze them. Chapter 4 summarizes the most important topic modeling concepts, presents the method used to create this thesis' topic model, and discusses some of the results obtained with the model. Chapter 5 does the same for the sentiment analysis. Finally, in chapter 6, I conclude and suggest further investigations.

Chapter 2

Background

In this section, I provide a brief overview of what NANOG is and what its mailing list is used for. Also, I review what a natural language processing workflow looks like.

2.1 The NANOG Mailing List

2.1.1 NANOG

NANOG stands for "North American Network Operators' Group." It is a group that provides services to the networking community and says that it is "providing a platform that inspires, educates, and empowers our community to meet the ever-changing demands of a global network, in service of building the Internet of tomorrow [5]." Those services include holding meetings with presentations and organizing other events and education to share networking best practices and Internet operations knowledge [5].

The predecessor of today's NANOG was established in 1987. It was a consortium responsible for re-engineering the high-speed Internet backbone of the "National Science Foundation Network." The enhanced backbone was to improve the connection of research and education facilities in the US. During the early times, the group organized "Regional Techs" meetings to gather and share networking information and talk about issues faced by internet service providers. Then over time, these meetings developed into an independent group that created their first charter in 1994 and later into the NANOG of today [4].

2.1.2 The Mailing List

The NANOG mailing list[1] is operated by NANOG. It is a place where network operators and researchers can post questions or routing reports, inform about outages, attacks, and vulnerabilities, recommend hardware and software, or talk about current topics. One might call it the social media of network operators. The mailing list's online archive[2] contains all emails sent since 1992 up to today. In total, it contains over 89,000 emails written by about 9,500 authors. These are the emails I will be analyzing in this report.

[1] `https://www.nanog.org/resources/nanog-mailing-list/nanog-mailing-lists/`
[2] `https://mailman.nanog.org/pipermail/nanog/`

2.2 NLP Workflow

Most natural language processing (NLP) projects share the same basic procedures. The prevailing approach is to first clean and preprocess the collected text data, get an overview of the data by doing some exploratory data analysis (EDA), and then apply different NLP techniques depending on the task [1].

Before text data can be processed, it must be cleaned according to how it is processed later and what the data looks like. One has to preprocess the text to help the machine learning algorithms used later to achieve better results. An algorithm treats all input terms with the same relevance and regards words differing by a single character as entirely different words. To facilitate the algorithm's processing of text, one has to normalize the text and remove noise.

Normalizing is usually done with the help of a library. One example of normalization is stemming. Stemming is a technique to remove suffixes from words to reduce them to their word stem. For example, "walk," "walking," "walks," and "walked" would all be turned into the word "walk". After reducing the words, any algorithm recognizes them as terms with the same meaning.

Noise comprises components of text data that do not help an algorithm to grasp its meaning. Removing noise includes, for example, deleting digits and special characters as well as topic-specific removals like HTML tags or email signatures. Typically, one also removes a list of so-called stop words, which consists of common words like "you," "if," or "have." Noise removal is often done manually, using regex patterns.

The normalization and noise removal examples are only a fraction of the possible measures to take; see section 3.2 for further cleaning and preprocessing steps. Often, there has to be another round of data cleaning with more steps after one has seen the result of the first analysis.

After cleaning the data, the next step is to do exploratory data analysis (EDA). EDA is employed to get an overview of the data, understand it better, make sure it makes sense, and help determine which NLP techniques to apply later. Commonly, EDA uses some types of visualization methods such as word clouds [2].

When the data has been appropriately preprocessed, and the EDA has given satisfactory results, one can apply NLP techniques. As there are many techniques to use, one has to choose one or try a few that fit the given task. Popular NLP tasks include topic modeling, sentiment analysis, or text generation, and all of them can be solved with different methods. Current NLP methods typically involve training a model using supervised or unsupervised machine learning or exploiting a downloaded, pre-trained model. Pre-trained models are useful because they were already trained with an appropriately large amount of data. Hence, one can save the time of training a model and also use them with datasets that would be too small to train a new model. However, using pre-trained models only works well if they were trained on text similar to one's own. If one wants to analyze text containing domain-specific language, a pre-trained model might not produce the best results, and training a new one might be more practical.

Chapter 3

Data Collection and Cleaning

This section explains how I collect the emails to put them in an appropriate data structure. I then present how I clean and preprocess the data for further analysis.

All mentioned scripts and a readme on how to use them can be found on GitLab[1].

3.1 Data Collection

For my data collection, I scrape the archive of the NANOG mailing list using a python script and the `requests` and `lxml` libraries. I then save the vital information in a `pandas` dataframe.

The emails in the archive are sectioned by month. Interestingly, there is a hole in the archive: Almost all emails from the year 2000 are missing inexplicably; there are only 25 emails from January 2000. To look at the emails, one can either download them as a per-month text file or view them online one email at a time. As information like thread titles and timestamps is more readily available when looking at the emails online, I decided to scrape them. This method's other advantage is that many of the "answered to" email parts are removed automatically because of how the HTML is structured.

To gather all facts about the emails, I use python's `requests` library to download the HTML and `lxml` to parse the HTML elements. I chose to put the scraped information in a `pandas` dataframe because it is a convenient data structure for data science purposes. In figure 3.1, there are all the columns I create and an example row of the dataframe.

id	thread_id	timestamp	level	thread_title	author	email_text
integer	integer	datetime	integer	string	string	string
212060	10962	2021-02-16 13:25:12+00:00	3	Texas internet connectivity dec..	Mike Hammett	It's cheaper to build 2x, 3x...

Figure 3.1: The structure of the dataframe with the raw emails showing all columns, data types, and an example row.

The `id` is a unique value for each email and is taken from the URL used when viewing an email. The `thread_id` is an ID I generated for each thread, so I know which emails belong together, and the `thread_title` is the subject line of the first email belonging to a thread. The `timestamp` is

[1] `https://gitlab.ethz.ch/nsg/students/projects/2021/sa-2021-09-mailing-list-analysis/-/tree/master/workspace`

the UTC at which the email was sent. The `level` of an email corresponds to its nesting level; if the email in question is a reply to a reply, for example, its level is 2. Lastly, the `author` is the writer of the email, and `email_text` is the actual email.

3.2 Data Cleaning

To later analyze or further preprocess the email texts, I first have to normalize some parts of the emails and remove noise. This data cleaning includes steps like removing punctuation and discarding email parts that do not add meaning. To clean the email texts, I mostly use `pandas`' `.replace()` function with regex patterns and substitute undesirable terms.

At this point in the NLP process, I normalize the emails primarily to simplify the noise removal. There are more reasons later when I specifically preprocess the text for the topic modeling, as explained in section 3.2.1. The first normalization step is to convert all text to lower case. I do this at the beginning because it simplifies the further replacements. The next step is to substitute common contractions like "isn't" with their expanded version. This substitution is necessary as I will remove apostrophes too, and otherwise, there would be words like "isn" and "haven." Lastly, I substitute IP addresses and AS numbers by "ipaddress" and "asnumber" as well as versions of "ipv4" and "ipv6" by "ipvfour" and "ipvsix". These expressions are relevant for the context of an email and should not be removed as they would be when I delete digits and punctuation; hence I replace these numbers with letters first.

Next, I have to remove the noise. The concept of noise encompasses all parts of the emails that are not conducive to the upcoming topic and sentiment analysis. There are email components like confidentiality notices not relevant for the content of the email that might even skew the results because the employed algorithms consider them as important as the other parts. Thus these parts should be removed. I start by removing "answered to" lines that were not already removed automatically during the scraping and email signatures, including confidentiality notices. Then, I delete "On <date> at <time> <person> wrote:" lines. The next step is to remove punctuation and special characters. While keeping punctuation and special characters was necessary to match the three previous patterns, most do not add value to further processing. I remove all of them except "!" as the sentiment analysis considers this at a later point. Also, I replace any type of whitespace character, such as line breaks by a single space, and delete all digits. Both digits and various types of whitespaces are used frequently but do not bear meaning concerning the topic or the sentiment of a text.

In figure 3.2, there is an example of an email (with `id` 68528) before and after it is cleaned.

Finally, after doing the above, I remove as many log dumps as possible. The largest issue with logs is that they sometimes contain many repetitions of a term, for example, "win" as an abbreviation of "windows," which distorts the sentiment analysis because some terms such as "win" will be perceived as positive terms and skew the resulting sentiment. I find log dumps by searching for repeated patterns. Because of everything else that I normalized and deleted, many dumps only consist of repeating phrases. In figure 3.3, there is an example showing why dump removal works. After all data cleaning except removing the logs, one can see the repeating patterns that are deleted during the log dump removal in subfigure 3.3b.

As an additional step, I discard any emails that are too short. Due to all of the above data cleaning, many words are removed from the emails. Thus, some documents contain less than 30 or even no characters at all. Examples of such emails are `not here either curtis`, `well said doug`, or `anyone have details`. I discard all of those documents because they are too short to add any value to an analysis.

```
Over the last decade, 19 states have made it illegal for municipalities
to own fiber networks -- encouraged largely, I am told, by Verizon and
other cable companies/MSOs[1].
Verizon, of course, isn't doing any new FiOS deployments, per a 2010
press release[2].
FCC Chair Tom Wheeler has been making noises lately that he wants the FCC
to preempt the field on this topic, making such deployments legal.
[...]
Cheers,
-- jra

--
Jay R. Ashworth                  Baylink
Designer                     The Things I Think                       RFC 2100
Ashworth & Associates               2000 Land Rover DII
St Petersburg FL USA      BCP38: Ask For It By Name!           +1 727 647 1274
```

(a) The scraped raw email with id 68528 before data cleaning.

```
over the last decade states have made it illegal for municipalities to own fiber
networks encouraged largely told by verizon and other cable companies msos verizon
of course is not doing any new fios deployments per press release fcc chair tom
wheeler has been making noises lately that he wants the fcc to preempt the field
on this topic making such deployments legal [...] cheers jra
```

(b) The email after data cleaning.

Figure 3.2: The email with id 68528 after it is scraped and after data cleaning.

```
   163.244.98.0/24 |    AS1 | BBNPLANET               | Dell Computer Corporati
  163.244.100.0/24 |    AS1 | BBNPLANET               | Dell Computer Corporati
   162.42.130.0/24 |    AS1 | BBNPLANET               | Arizona Public Service
   162.42.133.0/24 |    AS1 | BBNPLANET               | Arizona Public Service
   162.42.135.0/24 |    AS1 | BBNPLANET               | Arizona Public Service
```

(a) Excerpt of a type of report that was sent regularly to the mailing list before cleaning.

```
ipaddress asnumber bbnplanet dell computer corporati ipaddress asnumber bbnplanet
dell computer corporati ipaddress asnumber bbnplanet arizona public service
ipaddress asnumber bbnplanet arizona public service ipaddress asnumber bbnplanet
arizona public service
```

(b) Excerpt of a type of report that was sent regularly to the mailing list after all data cleaning except removing the logs.

Figure 3.3: An example of a log dump before and after data cleaning except removing the logs, showing why the log dump removal works.

3.2.1 Preprocessing Pipeline for Topic Modeling

Common and frequent words such as "very" or "well" carry much sentiment yet little topic-related information. For this reason, I use an extended preprocessing pipeline only for topic modeling to remove more terms. This extended pipeline removes common words that negatively impact the finding of coherent topics yet are important for sentiment analysis and thus cannot be removed by default. Additionally, I further normalize the emails by lemmatizing all words. Lemmatization requires an NLP library and does not lead to an improved sentiment analysis; hence I only do it for the topic modeling.

For the topic-specific preprocessing, I use the `spaCy`[2] library's processing pipeline as it is easily customizable and offers all desired functionalities. Its default language processing pipeline includes almost all necessary stages like a tokenizer to split up the text into tokens, a named entity recognizer, and a lemmatizer, and I can easily add an additional stage to remove stop words.

Before starting the pipeline, I add the email subject to process it together with the email text. As described in section 3.2, I have discarded the shortest emails; still, there are many short emails like `could not agree with you more` that do not offer any insight into what topic they are talking about. To improve the topic assignment of such emails, I add the subject line to the text that will be processed. For example, the subject line of the above email is `internic address allocation policy`. Adding the subject line helps assigning short texts to a topic yet does not affect longer emails much, as they contain many other words influencing the topic assignment.

The first text normalization step of the pipeline lemmatizes the emails. Lemmatization is similar to the stemming described in section 2.2. However, while stemming just cuts off the suffix without caring if the resulting word is a valid word, lemmatization uses morphological analysis to output the base form of a word. As an example, a stemmer turns the word "studying" to "study" and "studies" to the invalid word "studi" and a lemmatizer returns "study" for both. Lemmatization is crucial for the upcoming topic modeling because the modeling technique used is based on word frequencies; thus, it is essential for words with the same meaning to humans to consist of the same ASCII characters.

After `spaCy`'s lemmatizing pipeline stage, I move on to removing stop words. Because the topic modeling is based on word frequencies, not removing common words would lead to them appearing in each topic. In the first round, I remove all words that were part of `spaCy`'s stop word list. Then, after looking at the topic model that is generated with the processed emails, I iteratively identify more words to add to the list to include email-specific words like "hi," "sincerely," "fwd," and similar.

Lastly, I use `spaCy`'s entity recognition to delete all people's names it recognizes. Without removing names, some of the topics are based around an active author who signed all of their emails. To keep this from happening, I remove personal names. Unfortunately, the entity recognizer is not good at recognizing names in salutations and signatures. Consequently, I also have to add the names of frequently posting authors to the list of stop words to prevent them from appearing as part of different topics.

To conclude this section, I again give the email from section 3.2 in figure 3.4 to illustrate how an email looks before and after it is preprocessed for topic modeling.

[2] `https://spacy.io/`

```
over the last decade states have made it illegal for municipalities to own fiber
networks encouraged largely told by verizon and other cable companies msos verizon
of course is not doing any new fios deployments per press release fcc chair tom
wheeler has been making noises lately that he wants the fcc to preempt the field
on this topic making such deployments legal [...] cheers jra
```

(a) The email after the data cleaning but before the topic modeling preprocessing.

```
state illegal fiber network encourage largely tell verizon cable company msos
verizon course new fios deployment press release fcc chair make noise lately want
fcc preempt field topic make deployment legal [...] cheer muni fiber politic
```

(b) The email after the topic modeling preprocessing.

Figure 3.4: The email from section 3.2 with id 68528 before and after topic modeling specific preprocessing.

Chapter 4

Topic Modeling

In this section, I briefly present basic knowledge about topic modeling. Then, I describe the method used to create the topic model discussing the advantages of this method over others tested. Finally, I present the resulting topic model and an investigation of the evolution of topics over time.

A jupyter notebook explaining how to train the topic model called `createFinalDF.ipynb`, and one showing how to analyze the results called `finalAnalysis.ipynb` can be found on GitLab[1].

4.1 Background

Topic modeling is an unsupervised machine learning technique to identify topics in unstructured text data. A topic model outputs two things: words that correspond to the topics it found and for each topic a cluster of documents associated with it. The supervised variant of topic modeling is called topic classification, which requires at least some documents to be pre-labeled with topics for training. As the mailing list emails are unlabeled, I did not pursue topic classification further.

There are two main groups of topic modeling methods using different approaches; methods based on document embedding and methods based on statistics.

The first group includes libraries like `Top2Vec`[2], which I use in this thesis, and `BERTopic`[3]. They first map each document to a high-dimensional vector. To do the mapping, they use a document embedding model, either a downloaded pre-trained model or a newly trained one. Next, they cluster the document vectors with a clustering algorithm like HDBSCAN[4]. As clustering algorithms tend to perform poorly in high dimensions, dimensionality reduction techniques such as UMAP[5] may be applied beforehand. The clusters found by the clustering algorithm are then identified as the topics.

The second group includes the older, well-known statistical models LDA (Latent Dirichlet Allocation) and LSI (Latent Semantic Indexing), which use word frequencies and co-occurrences to estimate the probabilities for each word in a document to belong to a topic. As input, they need a list of words belonging to each document (bag of words) and the number of expected topics.

[1] `https://gitlab.ethz.ch/nsg/students/projects/2021/sa-2021-09-mailing-list-analysis/-/tree/master/workspace/spacyPipelines`

[2] `https://github.com/ddangelov/Top2Vec`

[3] `https://github.com/MaartenGr/BERTopic`

[4] `https://github.com/scikit-learn-contrib/hdbscan`

[5] `https://github.com/lmcinnes/umap`

4.2 Method

4.2.1 Adapted Top2Vec Library

To create the topic model for this thesis, I use the `Top2Vec` library, which I slightly adapt to my needs. `Top2Vec` builds embedded document vectors from the preprocessed input, and then after a dimensionality reduction, applies a clustering algorithm to find document clusters. From each document cluster, it finds the words belonging to the identified topic. The explanation of why I use this approach can be found in section 4.2.1.

The procedure in more detail is as follows: To construct the vectorized representation of the emails, I use `gensim`'s Doc2Vec model[6]. I train the embedding model on all preprocessed emails to enable it to infer a vector of dimension 300 for each document. Since the chosen clustering algorithm does not scale well to this many dimensions, the dimension reduction algorithm UMAP reduces the vectors to five dimensions. Then the clustering algorithm HDBSCAN is applied with a minimum cluster size of 40 because I was not interested in topics containing less than 40 emails. For each of the clusters found by HDBSCAN, `Top2Vec` calculates the centroid, which it uses as the topic vector. With the help of these topic vectors, it assigns precisely one topic to each email. The assignment is done by calculating the distance of the email's document vector to each topic vector and choosing the topic with the smallest distance. It is important to note that all emails are assigned a topic, even if the clustering algorithm classified them as an outlier.

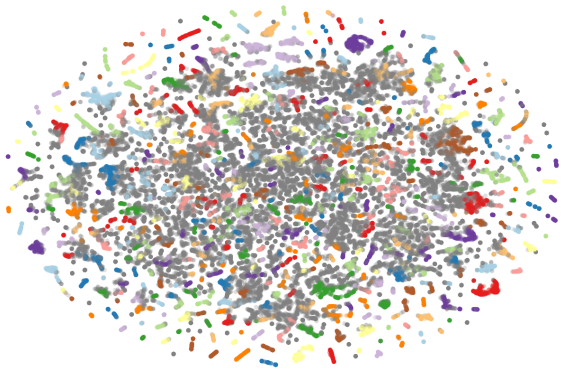
After the topic model is generated, the `Top2Vec` library offers many convenient utilities, among others, the function to merge topics to reduce them to a smaller number while keeping them meaningful.

Why Adapted Top2Vec?

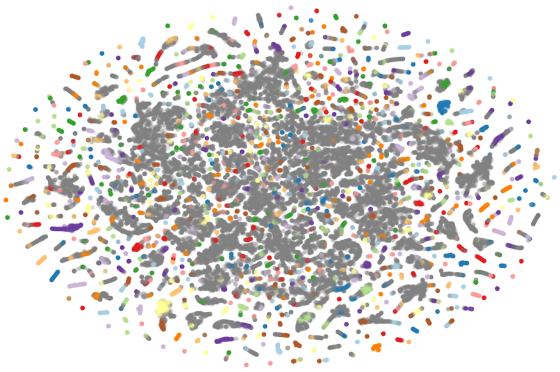
There are many reasons for using a slightly adapted version of the `Top2Vec` library. First of all, when I compared different methods by looking at the words belonging to each topic without spending too much time tuning each of them, it produced the best results. For example, it fared better than `gensim`'s[7] and `Mallet`[8] LDA, which are older, statistical topic modeling approaches (a result of gensim LDA with 30 topics can be found in appendix A). This difference might be because `Top2Vec` uses the newer document embedding technique to map documents to numerical vector spaces to then cluster these document vectors. Each cluster is identified as one topic. As a result, there is no need to guess the number of topics ahead of building the model as opposed to an LDA model, which requires the number of expected topics as an input. `Top2Vec` creates as many topics as needed and, if desired, reduces the number later on.

I found that the procedure applied by the `Top2Vec` library is well-chosen when I weighed it against others of the same group, particularly because it makes it possible to train an embedding model specifically for the email texts. For example, I also tested the `BERTopic` library, which is very similar to `Top2Vec`, with the difference being that `BERTopic` uses a pre-trained embedding model. `Top2Vec` offers the possibility to train one's own Doc2vec embedding model with `gensim`. Training my own embedding model improves the topic modeling substantially because the vocabulary used in the emails is very domain-specific, which is not handled well by pre-trained models. When training the model with Doc2vec, I assessed both a model trained where one document corresponded to one email and one where a document consisted of all emails of a thread. Both of them achieved similar

[6] `https://radimrehurek.com/gensim/models/doc2vec.html`
[7] `https://radimrehurek.com/gensim/models/ldamodel.html`
[8] `http://mallet.cs.umass.edu/`



(a) 2D Visualization of the 5D clusters found by HDBSCAN. Points of the same color belong to the same cluster and outliers are grey.



(b) 2D Visualization of the 5D clusters found by OPTICS. Points of the same color belong to the same cluster and outliers are grey.

Figure 4.1: Scatter plots of the emails' embedded 5D vectors after UMAP's dimensionality reduction. Each point in the plot corresponds to a 5D email vector, which is visualized in a 2D plane. Points close to each other with the same color belong to a cluster the clustering algorithm identified, and points the algorithm identified as outliers are grey.

results. However, as the sentiment analysis of chapter 5 works on a per-email basis, I decided to use the per-email embedding model.

For the next step, the dimensionality reduction, `Top2Vec` employs UMAP because it is a good choice for this use case. Compared to other dimensionality reduction algorithms, UMAP preserves more of the global structure, and the distances between clusters of points are meaningful [3], which benefits the subsequent clustering. Meaningful distances imply that nearby clusters belong to topics close in meaning. Compared to `Top2Vec`'s original implementation, I only slightly changed one of UMAP's parameters such that, according to UMAP's documentation, the result is better suited to be clustered, even though I did not observe it to make a big difference.

To cluster the document vectors, I decided to continue using the algorithm chosen by the library, HDBSCAN, as well. One benefit of HDBSCAN is that it identifies clusters of any form, no matter if they are spherical or shaped like crescent moons. Furthermore, it requires little user input since it is self-adjusting, and it classifies fewer points as outliers than the OPTICS[9] clustering algorithm. OPTICS tends to classify a large portion of the points as noise. The difference can be seen in figures 4.1b and 4.1a; the outliers, colored in grey, were more numerous when I used the OPTICS algorithm than when I used HDBSCAN. Compared to the default settings of `Top2Vec`, I changed HDBSCAN's parameters to increase the minimum cluster size because it is easier to analyze fewer but larger topics than many small ones.

The final reason for using the `Top2Vec` library is the convenience of the functions it already implements. The mentioned topic reduction is crucial, but also other functions like searching for topics by keywords or searching the documents by topic saved me the time it would have taken to create them myself.

[9] `https://scikit-learn.org/stable/modules/generated/sklearn.cluster.OPTICS.html`

4.3 Resulting Topic Model and Discussion

4.3.1 100-Topics Topic Model

The final topic model includes 454 topics. I tested different approaches to label the topics automatically, but they did not produce any usable results. Thus, I resorted to labeling them by hand. However, this required me to reduce the number of topics, as I did not want to spend too much time labeling. So, for the rest of the analysis, I merged the topics to 100. For the reduction, I used `Top2Vec`'s function that repetitively merges the smallest topic into the topic closest to it until 100 topics are left.

In figure 4.2, there is a word cloud representation of the first 15 topics titled with the manually picked topic label. The word clouds consist of the 50 words which have the highest semantic similarity to the topic vector. A higher similarity score leads to a larger word. All 100 hand-labeled topics can be found in appendix B. The largest topic is 1: "outages" and includes 2,464 emails, while the smallest one, topic 76: "email headers," includes 342 emails. To label the topics, I looked at the word clouds and at the emails closest to the topic according to the model.

Figure 4.2: Word cloud representation of first 15 topics of the 100 total topics.

Concerning the quality of the topics, almost all of them seem coherent and make sense by eye. However, there are a few topics that do not seem to have an easily determinable label. For example, topic number 34, which I labeled "mixed thread talk" or 85: "various requests and short emails." I could not find a fitting label for them because the emails were discussing various topics. Also, some topics like 1: "outages" and 34: "fiber cuts" are quite similar and could easily have been part of the same topic.

When looking at the emails assigned to each topic, the emails with a high similarity score to their topics appear to be well-matched. On the other hand, the emails with lower similarity scores frequently do not seem to fit the topic very well. However, most often, the lower-rated emails do not have a discernible topic at all. In figure 4.3, there is an example of an email with a high similarity score and one with a low score for topic 6: "DDOS attacks."

```
This appears to be a TCP amplification attack. Similar to UDP
amplification (DNS, NTP, etc) you can get some amplification by sending a
SYN packet with a spoofed source, and watching your victims receive
multiple SYN-ACK retries.  It's a fairly weak form of attack (as the
amplification factor is small), but if the victim's gear is vulnerable to
high packet rates it may be effective.
```

(a) Example email with a high similarity score to topic 6: "DDOS attacks".

```
I've done it, but if you've forced your way through my various filters
and manage to get me on the phone and I ask you that, it's pretty much
kiss of death unless you have a very good answer.
```

(b) Example email with a low similarity score to topic 6: "DDOS attacks".

Figure 4.3: Two emails with topic 6: "DDOS attacks" with a high, respectively low similarity score to the topic.

The problem might be that there is no notion of outliers in the topic model. Even though the clustering algorithm classifies some emails as outliers, as was seen in figure 4.1a, these outliers are just assigned the closest topic as well.

In conclusion, the topic model is not perfect, yet it is able to recognize meaningful topics in many of the emails.

4.3.2 Super-Topics

As I pointed out in section 4.3.1, a few topics are pretty similar, and additionally, many could be part of a larger, more comprehensive topic. For example, 36: "intradomain routing" and 9: "BGP metrics and announcements" could both be part of something like "routing protocols." Thus, it might make sense to add a hierarchy and find more comprehensive topics. I want to investigate if there are super-topics a level above the 100 topics.

To find these super-topics, I again use `Top2Vec`'s topic merging function to reduce the number of topics to nine. This merging function is very convenient and works well. I tried different numbers of super-topics, and nine is the number that produced the most sensible super-topics. Automatically determining an appropriate number of super-topics might be an interesting question for future work but is out of scope for this thesis.

One can see the word cloud representation of the super-topics in figure 4.4. I labeled them 0:

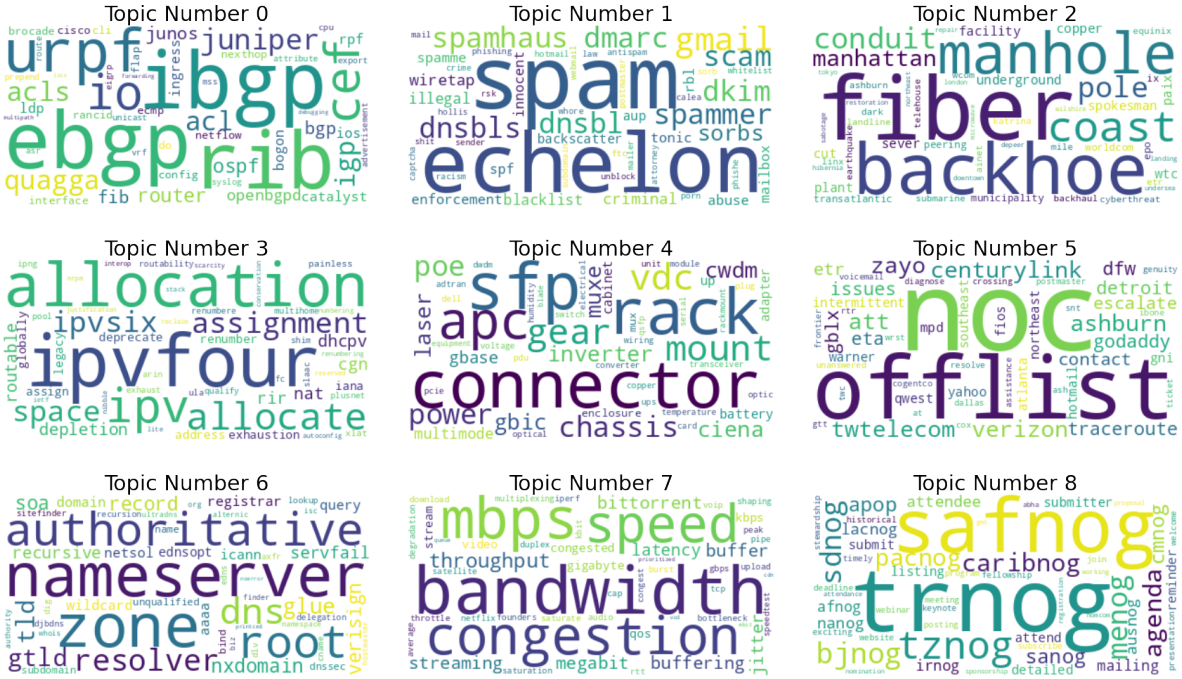


Figure 4.4: Word cloud representation of the 9 super-topics, named 0: "routing and protocols," 1: "spam and criminal activity," 2: "outages and providers," 3: "address space," 4: "server rooms," 5: "offlist contact," 6: "DNS," 7: "bandwidth and congestion," and 8: "network operator group."

"routing and protocols," 1: "spam and criminal activity," 2: "outages and providers," 3: "address space," 4: "server rooms," 5: "offlist contact," 6: "DNS," 7: "bandwidth and congestion," and 8: "network operator group." The largest one, super-topic 0: "routing and protocols" consists of 22,105 emails, and the smallest one, 4: "server rooms," of 3,713 emails. Topic 0: "routing and protocols" is also the super-topic for most of the 100 smaller topics. It is above 27 smaller topics. The topic with the least emails, topic 4: "server rooms," includes five smaller topics, and topic 5: "offlist contact" is above the least number of smaller topics with only four.

The list of which smaller topics belong to each super-topic can be found in appendix C. Looking at this list, one can see that the creation of the super-topics performed well. The classification of the smaller topics would have been similar if I had done it by hand.

4.4 Evolution of Topics Over Time

In this section, I show how I use the topic model to gain insights into what topics were discussed during which times.

4.4.1 Largest Topics During Period of Time

One question to answer is which topics were most important at which points in time. I implemented a function to output the n largest topics during specified years, where I define the size of a topic to be the number of emails assigned to it.

Considering the largest topics from 1992 – 2000 (figure 4.5), 2001 – 2010 (figure 4.6), and 2010 – 2021 (figure 4.7), one can see that the importance of topics has changed over time. I plotted each timespan's five largest topics over the whole 30 years.

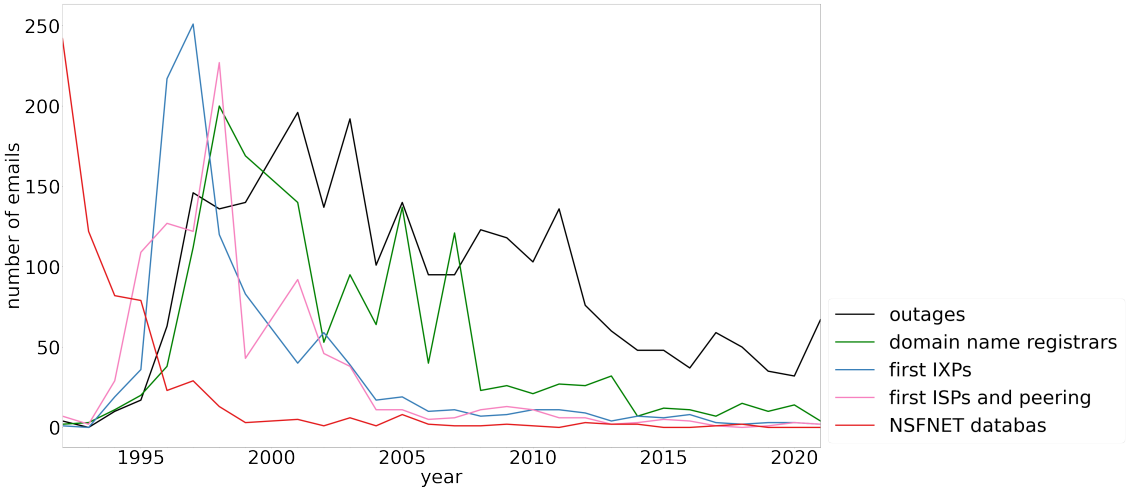


Figure 4.5: The size of the five largest topics during the period 1992 to 2000 plotted over all time.

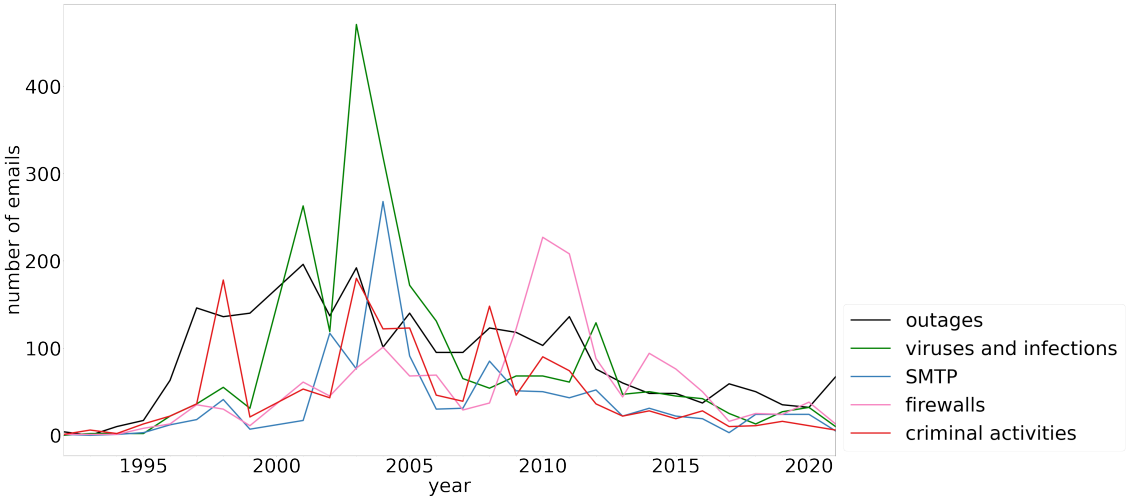


Figure 4.6: The size of the five largest topics during the period 2001 to 2010 plotted over all time.

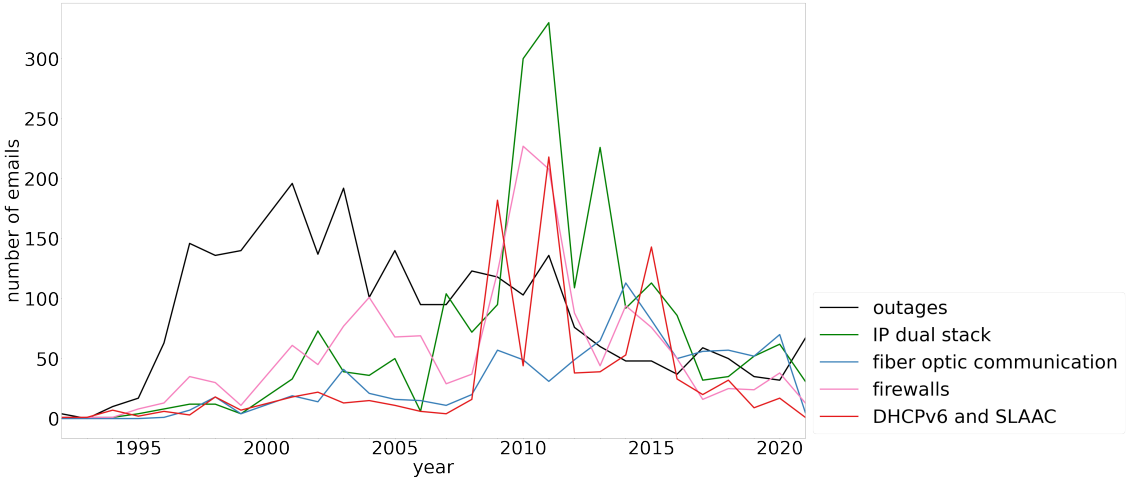


Figure 4.7: The size of the five largest topics during the period 2011 to 2021 plotted over all time.

These three plots already give some insight into the importance of different topics over time. Concerning the first years of the mailing list, one can see that it mainly consisted of "NSFNET database" reports, and only in 1994 and later people started to talk about other topics. Also, some topics were primarily important during the nineties and stopped being large later, like "first ISPs and peering" and "first IXPs." These topics mainly concern old Internet service providers and the first Internet exchange points. Some topics, namely "outages," were constantly talked about since the mailing list gained traction, while others exhibit distinct peaks (see section 4.4.2 below for more details). For example, there is a peak of "viruses and infections" and "criminal activities" around 2003, maybe because a few CISCO vulnerabilities were reported.

4.4.2 Peakiest Topics During Period of Time

As was seen in 4.4.1, some topics were very significant only for a short time, which gives interesting insights about the discussion of what was important at which point in time. I want to find these "peaky" topics.

To identify the peakiest topics, I first count how often each topic appears each year and exclude all topics with more than five emails every year after 1996. Excluding these topics helps to separate peaky topics from consistently important ones. Then, I apply the median absolute deviation (MAD) to each year's counts of the topics and return the topics with the largest MAD. MAD is very similar to the standard deviation, but it puts less weight on outliers.

I compared the results using median absolute deviation to the results using the standard deviation, and the median absolute deviation fit my use case better. I also tried to normalize the counts of topics per year to avoid constantly important topics with an even larger peak; however, this did not result in a better selection of peaky topics, and I discarded the idea.

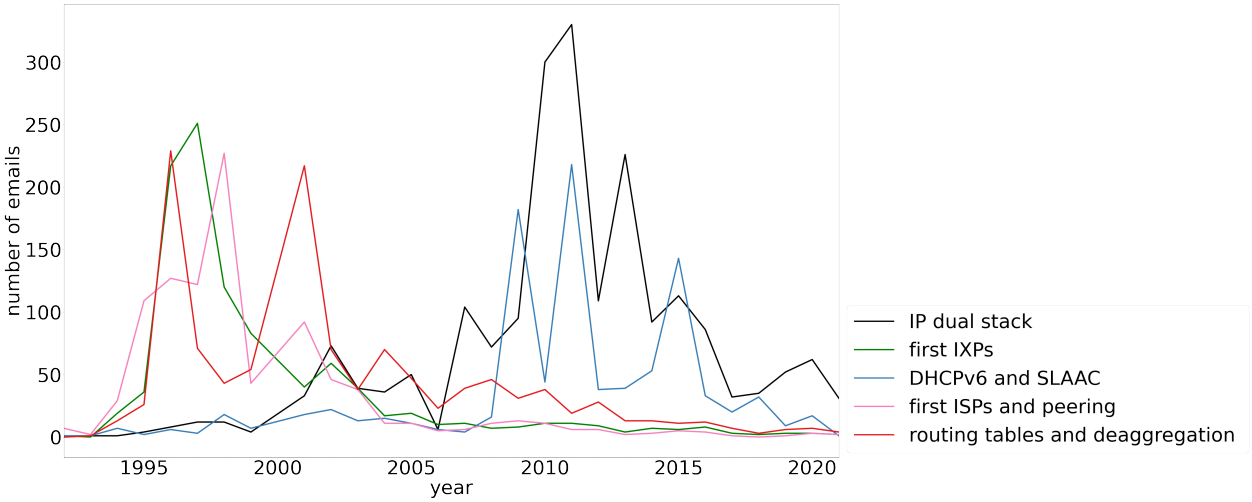


Figure 4.8: The five peakiest topics over all time according to my method.

In figure 4.8,one can see the resulting five peakiest topics. The largest peak is from the topic "IP dual stack" around 2011. The peak makes sense because IPv6 was launched in 2012, and it would seem reasonable that network operators were talking about it before and after the launch. The same holds for the IPv6 protocols topic "DHCPv6 and SLAAC," which peaks around the same time. Another distinct peak is in the "first IXPs" topic, when the first IXPs like MAE-west, MAE-east, and PAIX came up in the nineties and when the first ISPs emerged with a peak in the topic "first ISPs and peering."

Chapter 5

Sentiment Analysis

In this section, I give a brief overview of different sentiment analysis approaches. Then, I present how I used sentiment analysis for this thesis and finally show some of the results achieved with it.

5.1 Background

Sentiment analysis is a natural language processing technique to deduce from written text how the author felt. It either results in a sentiment category like positive, neutral, and negative or a sentiment score, typically ranging between -1 for very negative texts and +1 for very positive ones. There are two main approaches to sentiment analysis; rule-based and machine learning.

Rule-based sentiment analysis, which I use in this thesis, uses human-made rules to give a sentiment score to a text. The technique uses lexicons that contain words rated by their sentiment. It counts the number of positive and negative words and aggregates their sentiment scores to a final number, usually between -1 and +1. This approach takes into account some of the word order, for example, by increasing the sentiment of an adjective when it is preceded by "very" and inverting it when it is preceded by "not." Still, it is generally quite naive since it does not consider any other context or word order. Nevertheless, it is easy to use and does not require any training.

Machine learning techniques do not require any rules crafted by humans, as the algorithm will automatically learn how to detect sentiment. They typically produce more accurate results; however, to be trained, they need enough pre-labeled data, which might not be feasible. Unfortunately, while there are some pre-trained models like IBM Watson's[1], they all charge usage fees.

5.2 Method

For my sentiment analysis, I use VADER[2] to add a sentiment score between -1 and +1 to each email. VADER is a rule-based tool tuned explicitly to social media text. This is a good fit for my use case as a big part of the emails might be compared to short social media posts. About a third of the cleaned emails are shorter than Twitter's 280 character limit. VADER understands, for instance, acronyms like "rofl" or "lol" which is helpful as such expressions are also used in some emails.

I compared VADER's sentiment scores to TextBlob's[3], another rule-based sentiment analyzer, by looking at the emails with the largest difference in scores. Because VADER gave the more

[1] `https://www.ibm.com/cloud/watson-natural-language-understanding`

[2] `https://github.com/cjhutto/vaderSentiment`

[3] `https://github.com/sloria/TextBlob`

appropriate scores, I chose to use it for my analysis.

Considering the possibilities, VADER suits my needs best. Still, it is not perfect, and one has to consider the peculiarities of the mailing list emails by adapting some parts of VADER. Firstly, there are domain-specific words that either have a different meaning in everyday language or are not assigned any sentiment by VADER. Secondly, many automatically generated emails with technical content and measurements are assigned a positive sentiment by VADER but should be neutral.

VADER allows adding and removing words from the lexicon it uses to calculate the sentiment score. I exploit this to attune VADER to my use case of technical networking vocabulary and email jargon. For instance, I remove the sentiment from words like "appreciate" and "please," which were rated positively by VADER's lexicon. They are overused in emails and consequently do not convey positive feelings. Additionally, I remove terms that are connoted positively or negatively in everyday language but are neutral technical terms for network operators. This includes removing words like "support," which refers to technical support services offered by companies, "resolve," which concerns DNS resolvers, or "block," "deny," and "allow," which are associated with firewall rules. I also add negatively connoted technical terms that were not part of the lexicon before, including "congestion," "malware," "hacker," and "outage."

The other problem is the type of emails I call Internet report emails. These automatically generated emails contain, for example, a list of addition to a routing database or routing table excerpts. VADER assigns very positive scores to most of the Internet report emails. They get positive scores because some words, e.g., company names like "Merit" are repeated and are rated positively because of their day-to-day meaning. Even though I try to remove log dumps during the data cleaning, I did not find a way to remove all report parts. To improve the accuracy of the sentiment analysis, I manually give recurrent Internet report emails a neutral sentiment score of 0.

In conclusion, I implement a function that assigns a sentiment score of 0 to an email if it is a recurrent report and otherwise uses VADER's sentiment analyzer with the altered lexicon to generate the score. Unfortunately, even with the adaptions, the solution is still not optimal, as can be seen in the upcoming section 5.3.

5.3 Resulting Sentiment Scores and Discussion

The result of the VADER-based sentiment analysis is a sentiment score for each email. To get an overview, I look at some of the most positively and most negatively rated emails. An example of an email with a sentiment score of -0.9929 can be found in figure 5.1. This email has a very

```
[...]
I will shutdown a compromised account on my end, but that doesn't stop
ATT's infected subscriber from spamming 100 other servers using 100
other stolen credentials. I may also send an abuse report to ATT if they
have an infected machine trying to perform a dictionary attack or brute
force logins against my port 587 SMTP server. ATT's going to deal with
the abuse reports as cheaply as possible. If they receive enough, I have
no doubt they'll repeat past mistakes.
```

Figure 5.1: Example of an email with a negative sentiment score.

negative undertone and is correctly assigned a negative sentiment score. I also roughly agree with the sentiment scores of other emails I examined.

On closer examination, I realize that many of the most strongly rated emails have an above-average length. To investigate this further, one can look at the smoothed mean absolute value of the sentiment score for email texts with different lengths in figure 5.2.

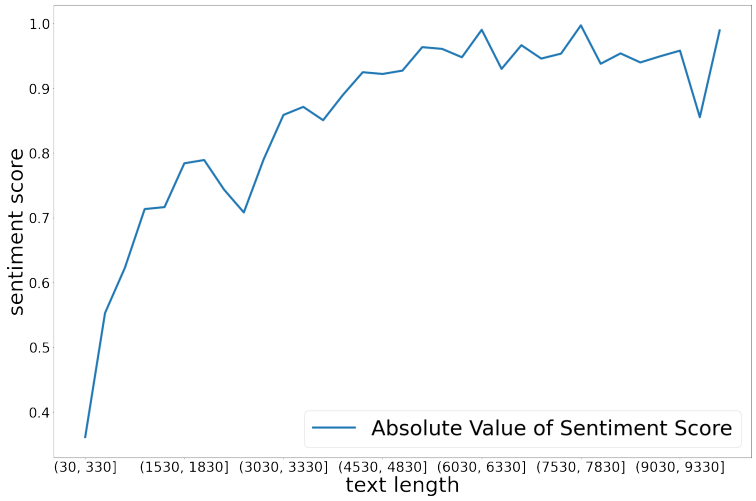


Figure 5.2: The smoothed mean absolute value of sentiment scores of increasing email text lengths.

It is evident from the figure that emails with longer texts are assigned more extreme sentiment scores. While one could say that people feeling more strongly about a subject will write longer texts about it, I believe that this phenomenon might be a limitation of VADER. Being created for social media text, VADER might not be able to deal with longer texts accurately. Also, when looking at the highly-rated long emails, I would often rate them only slightly positively or negatively. However, the long emails' sentiment scores do not affect the big picture much because there are only about 1,000 emails with more than 3,000 characters out of 89,000.

To conclude, while VADER has its limitations and might not be the optimal solution, it works reasonably well, especially for shorter emails.

5.4 Further Sentiment Analysis

In this section, I present different sentiment aspects I explored. This includes very positive, negative, and controversial threads and topics, as well as sentiments related to different keywords.

5.4.1 Sentiment of Threads

I want to explore the email threads with the most positive, most negative, and most controversial feelings. Each thread consists of three emails on average, but the number of emails per thread ranges from one to a few hundred. For each thread, I calculate the mean and the standard deviation of the sentiment score of all its emails.

To identify the most positive, negative, and controversial threads, I use these computed values and return the threads with the highest, respectively lowest mean, or for controversial threads the ones with the highest standard deviation in the sentiment score. Because many threads contain only one email, my function allows specifying the minimum number of emails a thread needs to have to be considered.

In figure 5.3, there is an example of the emails of a controversial thread with a minimum of two emails. Barry's email was correctly assigned a very low sentiment score, while Randy's good-natured answer gets a much more positive score.

5.4.2 Sentiment of Topics

I am interested in how people felt about different topics over the years. To answer this question, I examine how the sentiment about specific topics changed over the years, and I find the most positive, negative, and controversial topics in given periods.

To find out how network operators' sentiment about different topics evolved over the years, I calculate the mean sentiment of all emails belonging to a topic for each year. Figure 5.4 shows how the mean sentiment of topic 6: "DDOS attacks" emails developed over the years. Unfortunately, this plot does not give any interesting insights, even when looking at sentiment peaks, as no special events happened in those years.

To find the most positive, negative, and controversial topics, I use the same approach as for the threads in section 5.4.1. I calculate the mean and the standard deviation of the sentiment score of all emails belonging to a topic. If "positive" or "negative" is selected as the feeling, the topics with the highest, respectively lowest mean, are returned. If one wants controversial topics, the function returns the topics with the highest standard deviation in the sentiment score.

For example, the four most positive topics over the whole period are "elections," "measurements and surveys," "NANOG meeting logistics," and "call for presentations." These topics seem to make sense as candidates for a NANOG election want to present themselves in a good light, a call to complete a survey has to be written positively to get people to participate, people generally look forward to going to a NANOG meeting, and trying to get people to submit presentations also has to be written in a positive way. The most negative topics are those concerning cyber attacks like DDOS or viruses, which is also sensible since those usually are a problem and not something positive.

5.4.3 Sentiment of Topic Keywords

Often, one does not want to look through all topics to find the one that best fits the subject one wanted to plot the sentiment of, but use a keyword representing a topic. To make it easier to find network operators' sentiment about subjects, I want to make it possible to look at the sentiment of topics that fit best to one or multiple given keywords.

To plot the sentiment of the topics most semantically similar to the combination of given keywords, I first find the topics most similar to the given keywords. For this, I use `Top2Vec`'s built-in `search_topics` function. This function uses the document embedding model of section 4.2.1's topic model to map the keywords to a vector. Then it searches the model's topic vectors to return the topic, whose topic vector has the smallest distance to the vector generated from the keywords. When I have the topics closest to the keywords, I use the same method as in section 5.4.2 to examine the topics' sentiment development, namely calculating the mean sentiment score of each topics' emails for each year.

It is possible to look at only the most semantically similar topic but also to look at the two or three closest topics. If multiple topics are considered, I will look at the mean sentiment score of the emails belonging to all considered topics.

In figure 5.5, one can see how the sentiment of the two topics closest to the keyword "terrorism" changed over time. The two closest topics are topic 27: "cybersecurity and terrorism" and topic 85: "invasion of privacy." The sentiment fluctuations in the early years are due to very few emails

```
3PM EST and that mailbombing from a Sprint customer continues
full-blast, well into its third full day, and Sprint continues to
refuse to do anything about it.

Go ahead, tell me again how this is nothing but a spam whine -- three
days, thousands and thousands of messages, all being bounced back,
non-stop, and Sprint refuses to lift a finger or acknowledge that this
is an operational problem.

Sprint should not be allowed to operate on the internet, they're
destroying it, they're completely irresponsible.

--
        -Barry Shein [...]
```

(a) The first email of the thread has a very negative sentiment score.

```
Hi Barry,

For a friend, sure.
Barry, it's nothing but a spam whine.

But more seriously.
>>>
Well, it's not really an Ops problem.  Ops folk worry about packets and
stuff like that, not social and legal issues such as this.  Within
organizations such as Sprint, they're usually not empowered to take the
kinds of actions you want.

And you seem to have the attack effectively repelled for the moment.  So a
more cautious but longer lasting approach might be worth considering.  E.g.
Sprint management might need some constructive help to get an anti-spam
policy into their customer contracts.

[...]

Provider community management perceptions are driven by customer demand.
So it would be very helpful for the user community to make clear to big
provider management that they don't want to receive spam.

randy
```

(b) The second email of the thread has a very positive sentiment score.

Figure 5.3: Controversial thread with two emails. The first email is rated very negatively and the second one very positively.

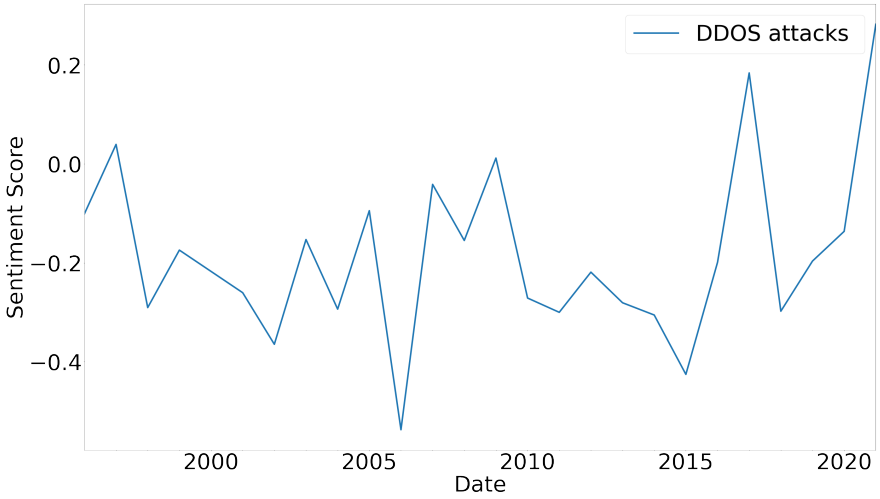


Figure 5.4: Average sentiment score of emails belonging to topic 6: "DDOS attacks".

assigned to the topics back then. Because there are only very few emails, a single more positively rated email strongly influences the yearly mean sentiment score. Still, there is an apparent dip in 2001 when people felt more extremely negative, probably after 9/11.

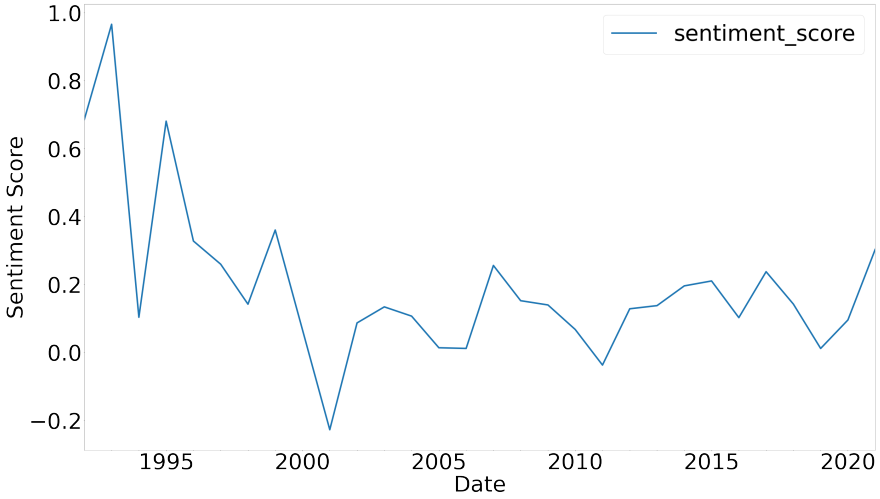


Figure 5.5: Average sentiment score of emails belonging to the two topics closest to "terrorism".

5.4.4 Sentiment of General Keywords

The last thing I examine is how network operators' sentiments about different companies, hardware manufacturers, ISPs, and similar, changed. I want to determine if it is possible to discern years where, e.g., companies messed up with a manufacturing error or ISPs had an outage. For this, I look at the sentiment score of emails containing the selected companies or brand names.

Because I am interested in specific brands only, the method from section 5.4.3 is inadequate. For example, different hardware brands are all mixed in topic "hardware models and brands"; hence, they cannot be distinguished when looking at topics. To make the sentiment more fine-grained, I calculate each year's mean of the sentiment score of all emails containing the given name or

keyword. This way, I can also directly compare different brands.

As an example, I compare the sentiment about different ISPs in figure 5.6. First off, it is interesting to see when the different ISPs came up; evidently, Comcast is one of the oldest network providers. Concerning the mean sentiment scores over the years, one can see fluctuations. Some fluctuations are due to hardly any emails being written in some years. Because there were very few emails, the emails' authors' opinion is weighted heavily, and it is less of a general view of network operators. However, other fluctuations like the dip of the ISP CenturyLink in 2015 are more likely because of major outage issues the provider faced in that year, leading to general dissatisfaction.

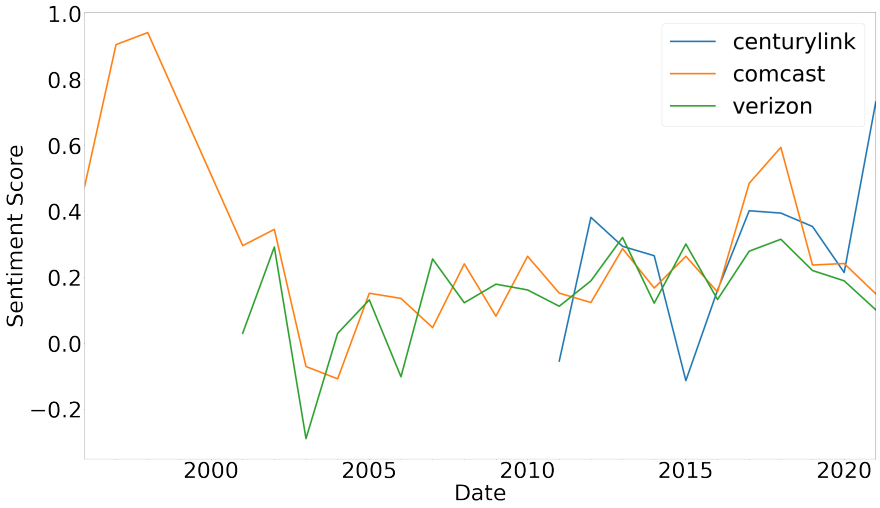


Figure 5.6: Average sentiment score of emails containing the words "centurylink", "comcast", or "verizon".

Chapter 6

Conclusion and Future Work

In this thesis, I looked at applying natural language processing techniques to the NANOG mailing list. After preprocessing the emails, I have created a topic model and conducted topic and sentiment analyses. For both the topic modeling and the sentiment analysis, as well as the preceding data cleaning, there are more aspects one could have looked at and elements that could have been implemented differently.

One detail to look at more closely is the dimensionality reduction during the creation of the topic model. In my procedure, first, every email is mapped to a 300-dimensional vector with gensim's Doc2vec model, and then the dimensionality reduction algorithm UMAP is used to reduce the vectors to five dimensions. It is also possible to map the documents to vectors with lower dimensions such that fewer dimensions have to be shaved off by UMAP. It would be interesting to investigate how a Doc2Vec embedding model using lower-dimensional vectors affects the topic model.

Another change offering more opportunities would be to assign more than one topic to each document, like topic tags. Often emails belong to more than one topic; for example, when an email talks about a vulnerability in routing software, it is part of the "routing software" topic and the "vulnerabilities" topic. If an email is assigned multiple topics, one might be able to draw connections between the different topics.

For the sentiment analysis, it might have been better to choose a different approach. As seen in section 5.3, the sentiment analysis works for many emails, yet there are some limitations to VADER. One can think about using a pre-trained tool like IBM Watson, which takes some more time to integrate but might give better results, or even spend some time labeling emails and training a new sentiment analysis model.

What might improve both the sentiment and topic analysis is to clean the emails more carefully beforehand. Even after removing some of the log dumps, there are still many emails with parts such as routing table excerpts, traceroutes, DNS responses, or IRR objects. One could think about a way to more rigorously remove these parts because they generally do not improve the topic model and tend to skew the sentiment analysis.

Bibliography

[1] Dipanjan (DJ) Sarkar. A Practitioner's Guide to Natural Language Processing (Part I) — Processing Understanding Text. https://towardsdatascience.com/a-practitioners-guide-to-natural-language-processing-part-i-processing-understanding-text-9f4abfd13e72, 2018. [Online; accessed 22-May-2021].

[2] IBM Cloud Education. Exploratory Data Analysis. https://www.ibm.com/cloud/learn/exploratory-data-analysis, 2020. [Online; accessed 22-May-2021].

[3] Mcinnes, L., Healy, J., and Melville, J. Umap: Uniform manifold approximation and projection for dimension reduction, 2020.

[4] NANOG. Our Story. https://www.nanog.org/about/our-story/, 2021. [Online; accessed 22-May-2021].

[5] NANOG. What we do. https://www.nanog.org/about/what-we-do/, 2021. [Online; accessed 22-May-2021].

Appendix A

LDA Model with 30 Topics

```
   [(0,
  [('vnpt', 0.08834264),
   ('net', 0.06647149),
   ('group', 0.06394054),
   ('communication', 0.051572442),
   ('vietel', 0.046278708),
   ('telkomnet', 0.041369725),
   ('esp', 0.037618943),
   ('mpls', 0.03715529),
   ('cmnet', 0.036305197),
   ('kixs', 0.03209171)]),
 (1,
  [('route', 0.052788444),
   ('prefix', 0.042547707),
   ('rpki', 0.034623917),
   ('bgp', 0.032236524),
   ('community', 0.020032221),
   ('invalid', 0.018095344),
   ('asn', 0.017598957),
   ('object', 0.016748045),
   ('filter', 0.016169367),
   ('asnumber', 0.014402176)]),
 (2,
  [('cable', 0.3685675),
   ('modem', 0.139298),
   ('cmts', 0.058886208),
   ('docsis', 0.04335466),
   ('finalize', 0.022850953),
   ('buildingnew', 0.0012926083),
   ('flix', 1.2231628e-05),
   ('therebetter', 1.2231628e-05),
   ('tray', 1.2231628e-05),
   ('twit', 1.2231628e-05)]),
 (3,
  [('port', 0.09115896),
   ('switch', 0.066666074),
   ('card', 0.034577236),
   ('cpu', 0.03410522),
   ('vendor', 0.024704533),
   ('support', 0.022655882),
   ('cisco', 0.022294713),
   ('ethernet', 0.02179234),
   ('cheap', 0.020622823),
   ('optic', 0.020497944)]),
 (4,
  [('level', 0.10613264),
   ('cci', 0.09252536),
   ('cxa', 0.07301827),
   ('rdc', 0.06944371),
   ('enterprise', 0.042646717),
   ('business', 0.034131065),
   ('cogent', 0.031468395),
   ('bezeqint', 0.03024207),
   ('linkdotnet', 0.023179993),
   ('reserved', 0.022992862)]),
 (5,
  [('inc.', 0.121666506),
   ('ovh', 0.11719833),
   ('networks', 0.10935064),
   ('llc', 0.075861834),
   ('telecom', 0.068642795),
   ('services', 0.064342685),
   ('gmbh', 0.041465536),
   ('media', 0.035956066),
   ('company', 0.025792768),
   ('ltd.', 0.025645)]),
 (6,
  [('mar', 0.43349963),
   ('host', 0.15006137),
   ('nov', 0.12572362),
   ('best', 0.014261927),
   ('wrst', 0.0118935555),
   ('stdev', 0.0118935555),
   ('fri', 0.011244089),
   ('avg', 0.010597048),
   ('mtr', 0.009720613),
   ('tangible', 0.0067220633)]),
 (7,
  [('traffic', 0.034656763),
   ('network', 0.020910751),
   ('attack', 0.019043272),
   ('source', 0.015287217),
   ('server', 0.01308773),
   ('udp', 0.012209335),
   ('block', 0.011856882),
   ('port', 0.011801573),
   ('customer', 0.011727901),
   ('ddos', 0.010179058)]),
 (8,
  [('ipaddress', 0.045964856),
   ('list', 0.02760296),
   ('issue', 0.02522272),
   ('contact', 0.017876973),
   ('time', 0.015248365),
   ('email', 0.013637642),
   ('look', 0.012960237),
   ('help', 0.011782541),
   ('alert', 0.011278989),
   ('network', 0.010966257)]),
 (9,
  [('peer', 0.16373052),
   ('cogent', 0.10374455),
   ('transit', 0.10251812),
   ('traffic', 0.06649549),
   ('provider', 0.054054745),
   ('network', 0.035027444),
   ('exchange', 0.031136053),
   ('internet', 0.028882328),
   ('customer', 0.025744502),
   ('connect', 0.01659121)]),
 (10,
  [('key', 0.12047938),
   ('pgp', 0.08548274),
   ('security', 0.06526905),
   ('hash', 0.06246749),
   ('sign', 0.030179473),
   ('begin', 0.029095212),
   ('signature', 0.027792485),
   ('affected', 0.027083732),
   ('fingerprint', 0.0262914),
   ('malware', 0.02590795)]),
 (11,
  [('san', 0.11363912),
   ('francisco', 0.07364826),
   ('labs', 0.06291918),
   ('onsite', 0.046573583),
   ('optimization', 0.03819481),
   ('austin', 0.03277529),
   ('downtown', 0.02920409),
   ('itechgeek', 0.013406531),
   ('keys', 0.0060862354),
   ('itg', 0.0034302962)]),
 (12,
  [('device', 0.022217238),
   ('run', 0.017983656),
   ('system', 0.017457575),
   ('work', 0.016452821),
   ('server', 0.014611075),
   ('like', 0.012544674),
   ('software', 0.011745136),
   ('support', 0.011016809),
   ('need', 0.010849594),
   ('look', 0.010740817)]),
```

```
(13,
 [('message', 0.04908423),
  ('information', 0.039601088),
  ('intend', 0.024148928),
  ('new', 0.022524813),
  ('contain', 0.020949142),
  ('email', 0.01939659),
  ('mail', 0.01784839),
  ('content', 0.016722402),
  ('sender', 0.015927935),
  ('error', 0.014392008)]),
(14,
 [('nanog', 0.049743827),
  ('presentation', 0.018415324),
  ('slide', 0.016599506),
  ('community', 0.0148945255),
  ('network', 0.012897503),
  ('draft', 0.012563418),
  ('committee', 0.012143882),
  ('technology', 0.011646733),
  ('submit', 0.0113103),
  ('topic', 0.01113636)]),
(15,
 [('ipaddress', 0.19269751),
  ('region', 0.04260755),
  ('announce', 0.031948734),
  ('routing', 0.027253168),
  ('table', 0.027099129),
  ('address', 0.026361851),
  ('asn', 0.025773704),
  ('ases', 0.024681333),
  ('internet', 0.024490662),
  ('prefix', 0.021871977)]),
(16,
 [('content', 0.09446927),
  ('netflix', 0.06390775),
  ('isp', 0.0634692),
  ('jul', 0.057669748),
  ('user', 0.051833957),
  ('stre', 0.04483054),
  ('cdn', 0.0396782),
  ('video', 0.034429587),
  ('traffic', 0.03346304),
  ('watch', 0.028375654)]),
(17,
 [('systems', 0.15392275),
  ('web', 0.08219343),
  ('temperature', 0.078098185),
  ('heat', 0.067178644),
  ('cabinet', 0.058993712),
  ('dirtside', 0.046666987),
  ('air', 0.041777514),
  ('owner', 0.031608123),
  ('dish', 0.02251477),
  ('vac', 0.012024382)]),
(18,
 [('ipaddress', 0.061547987),
  ('packet', 0.04016886),
  ('test', 0.02411944),
  ('connection', 0.013809931),
  ('issue', 0.013426807),
  ('tcp', 0.012313323),
  ('drop', 0.012214639),
  ('send', 0.010701833),
  ('receive', 0.00905491),
  ('loss', 0.009032875)]),
(19,
 [('service', 0.024862397),
  ('power', 0.019620938),
  ('fire', 0.016657094),
  ('customer', 0.013680982),
  ('fiber', 0.012746982),
  ('network', 0.010933284),
  ('building', 0.009083135),
  ('cost', 0.0085729575),
  ('mark', 0.0074791834),
  ('internet', 0.007380247)]),
(20,
 [('traceroute', 0.11296086),
  ('radb', 0.080908276),
  ('ncc', 0.076456234),
  ('vlan', 0.06277857),
  ('hop', 0.05628904),
  ('reclaim', 0.04942681),
  ('.net', 0.03700614),
  ('mnt', 0.02797226),
  ('.ms', 0.027779913),
  ('max', 0.024197167)]),
(21,
 [('signature', 0.12593241),
  ('byte', 0.09912523),
  ('size', 0.09775702),
  ('text', 0.09765991),
  ('desc', 0.09065434),
  ('type', 0.08884539),
  ('non', 0.08772553),
  ('ipaddress', 0.060980693),
  ('available', 0.040341116),
  ('signature.asc', 0.03696702)]),
(22,
 [('router', 0.03370433),
  ('route', 0.029640386),
  ('bgp', 0.02619875),
  ('mpls', 0.021660661),
  ('mark', 0.021603255),
  ('network', 0.020753533),
  ('routing', 0.016580794),
  ('link', 0.01456977),
  ('table', 0.0141273895),
  ('traffic', 0.012655576)]),
(23,
 [('icann', 0.17247055),
  ('beckman', 2.7670092e-05),
  ('abovebelow', 2.7670092e-05),
  ('therebetter', 2.7670092e-05),
  ('topair', 2.7670092e-05),
  ('tray', 2.7670092e-05),
  ('slurp', 2.7670092e-05),
  ('ahhhh', 2.7670092e-05),
  ('twit', 2.7670092e-05),
  ('snail', 2.7670092e-05)]),
(24,
 [('arin', 0.11391395),
  ('policy', 0.039970424),
  ('rir', 0.039026443),
  ('resource', 0.027896749),
  ('ripe', 0.023022944),
  ('registration', 0.022309389),
  ('registry', 0.020725787),
  ('register', 0.019575553),
  ('speaker', 0.019326698),
  ('organization', 0.016396046)]),
(25,
 [('ipv', 0.17800206),
  ('address', 0.060750425),
  ('customer', 0.022068033),
  ('space', 0.020303225),
  ('network', 0.018427664),
  ('support', 0.017958874),
  ('nat', 0.01663257),
  ('rfc', 0.015006225),
  ('need', 0.014864255),
  ('block', 0.013185805)]),
(26,
 [('information', 0.029635608),
  ('report', 0.029227532),
  ('law', 0.026419358),
  ('company', 0.02568708),
  ('address', 0.023420453),
  ('block', 0.02225852),
  ('legacy', 0.022135869),
  ('public', 0.021861292),
  ('government', 0.017548166),
  ('internet', 0.017182829)]),
(27,
 [('dns', 0.12989023),
  ('server', 0.086550176),
  ('domain', 0.0587185),
  ('record', 0.04276425),
  ('query', 0.04243932),
  ('whois', 0.029904712),
  ('section', 0.022148976),
  ('zone', 0.02155533),
  ('dig', 0.019503124),
  ('answer', 0.019329987)]),
(28,
 [('ixp', 0.2413477),
  ('ixps', 0.045467593),
  ('participant', 0.039850473),
  ('lan', 0.031354316),
  ('exchange', 0.0038271402),
  ('fabric', 0.0019368848),
  ('arp', 2.1276705e-05),
  ('beckman', 2.1276703e-05),
  ('tray', 2.1276703e-05),
  ('ahhhh', 2.1276703e-05)]),
(29,
 [('like', 0.010129463),
  ('thing', 0.009713118),
  ('people', 0.009192791),
  ('think', 0.008166125),
  ('time', 0.0076821093),
  ('know', 0.0072967703),
  ('way', 0.0067191278),
  ('work', 0.006654675),
  ('say', 0.0065977634),
  ('case', 0.006464312)])]
```

Appendix B

All Topics

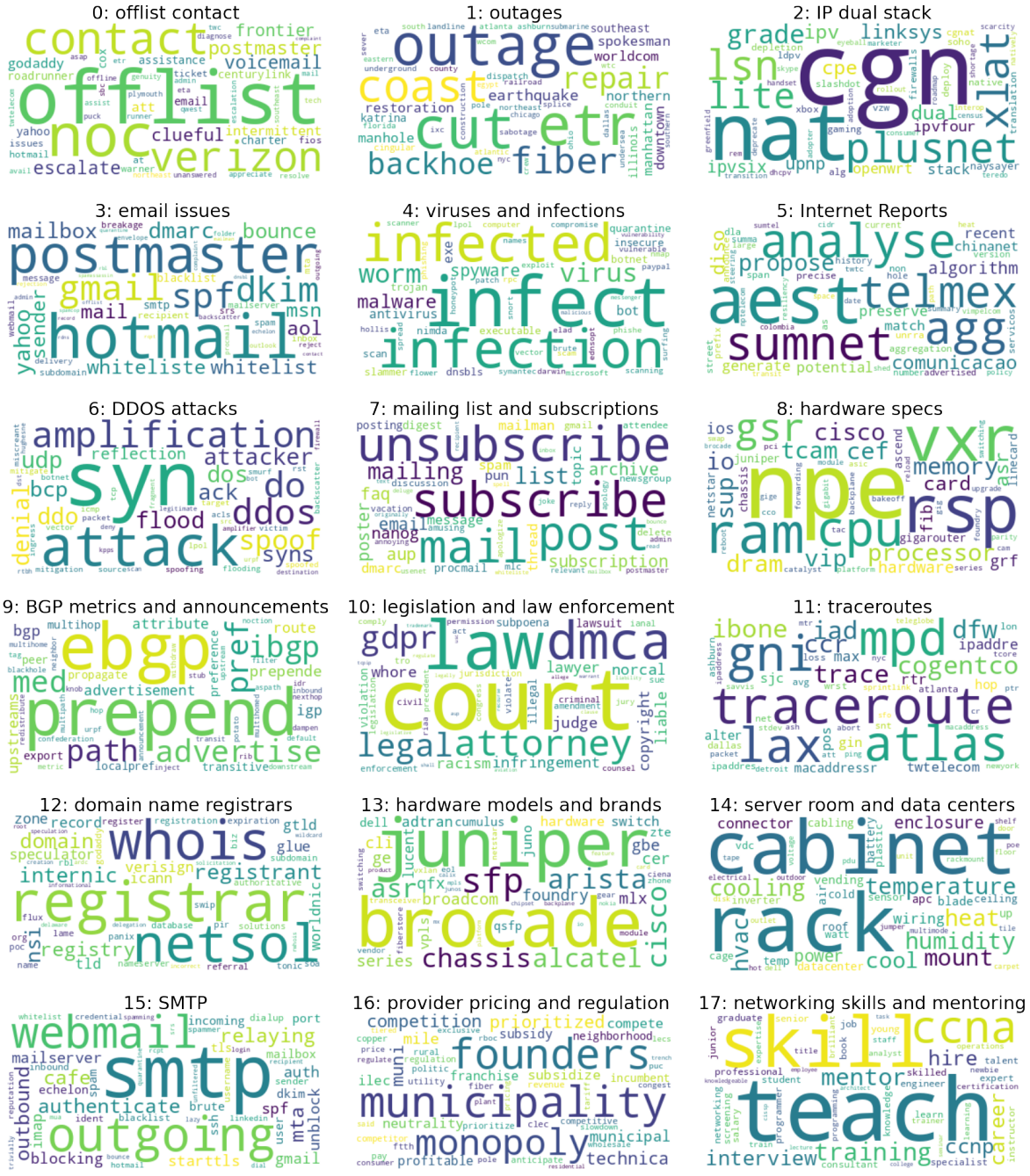


Figure B.1: Wordcloud representation of topic numbers 0 to 17.

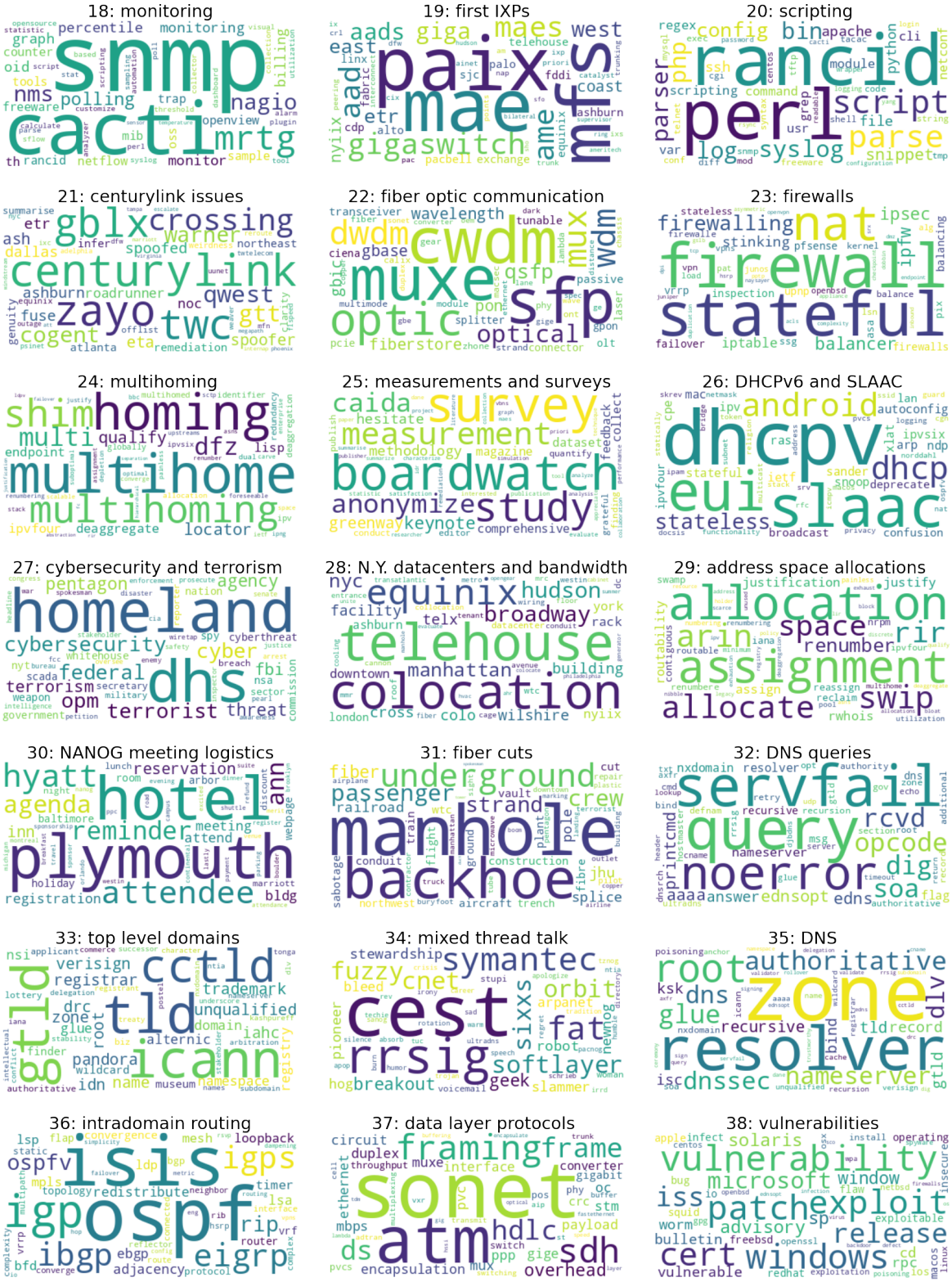


Figure B.2: Wordcloud representation of topic numbers 18 to 38.

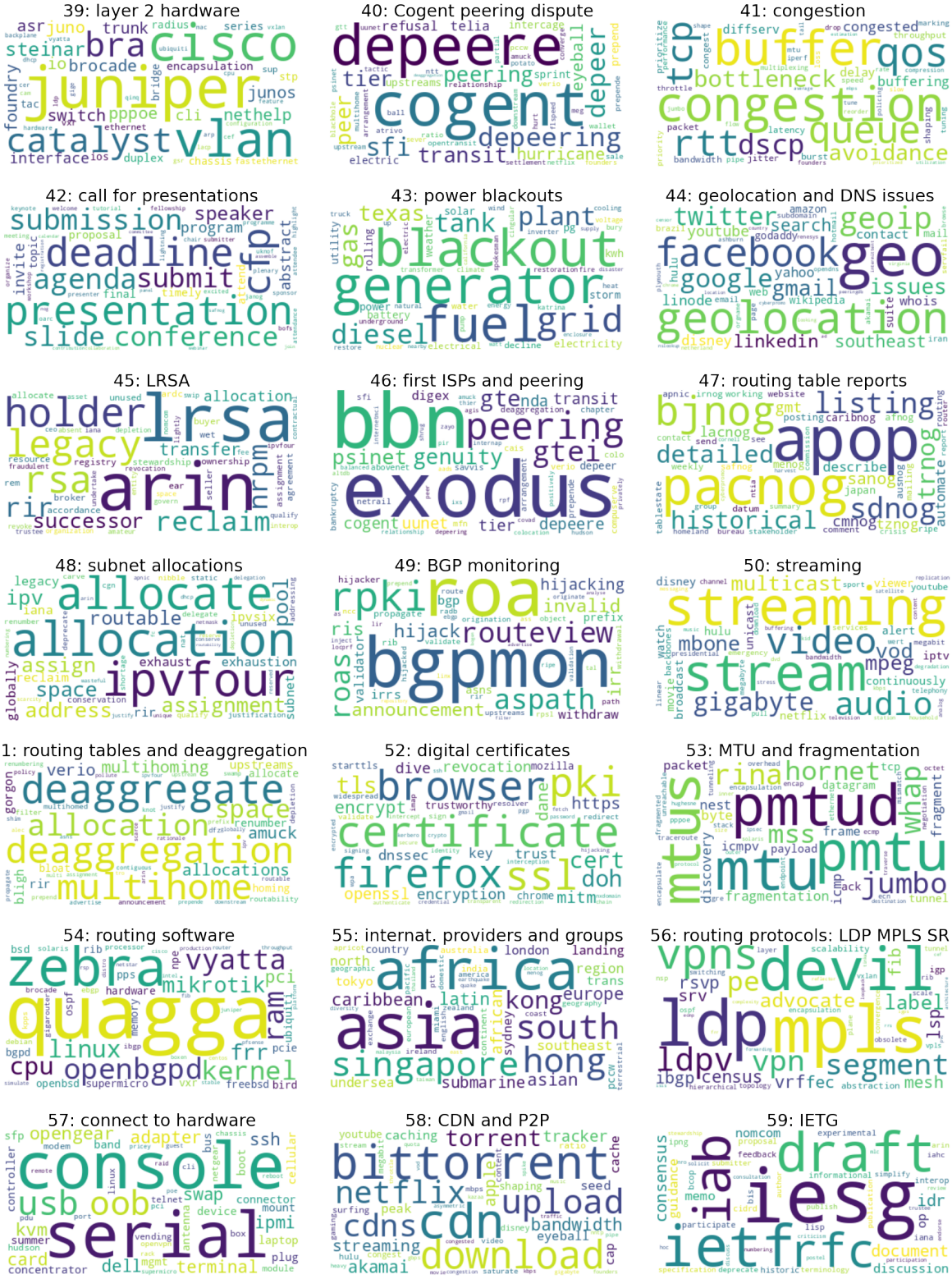


Figure B.3: Wordcloud representation of topic numbers 39 to 59.

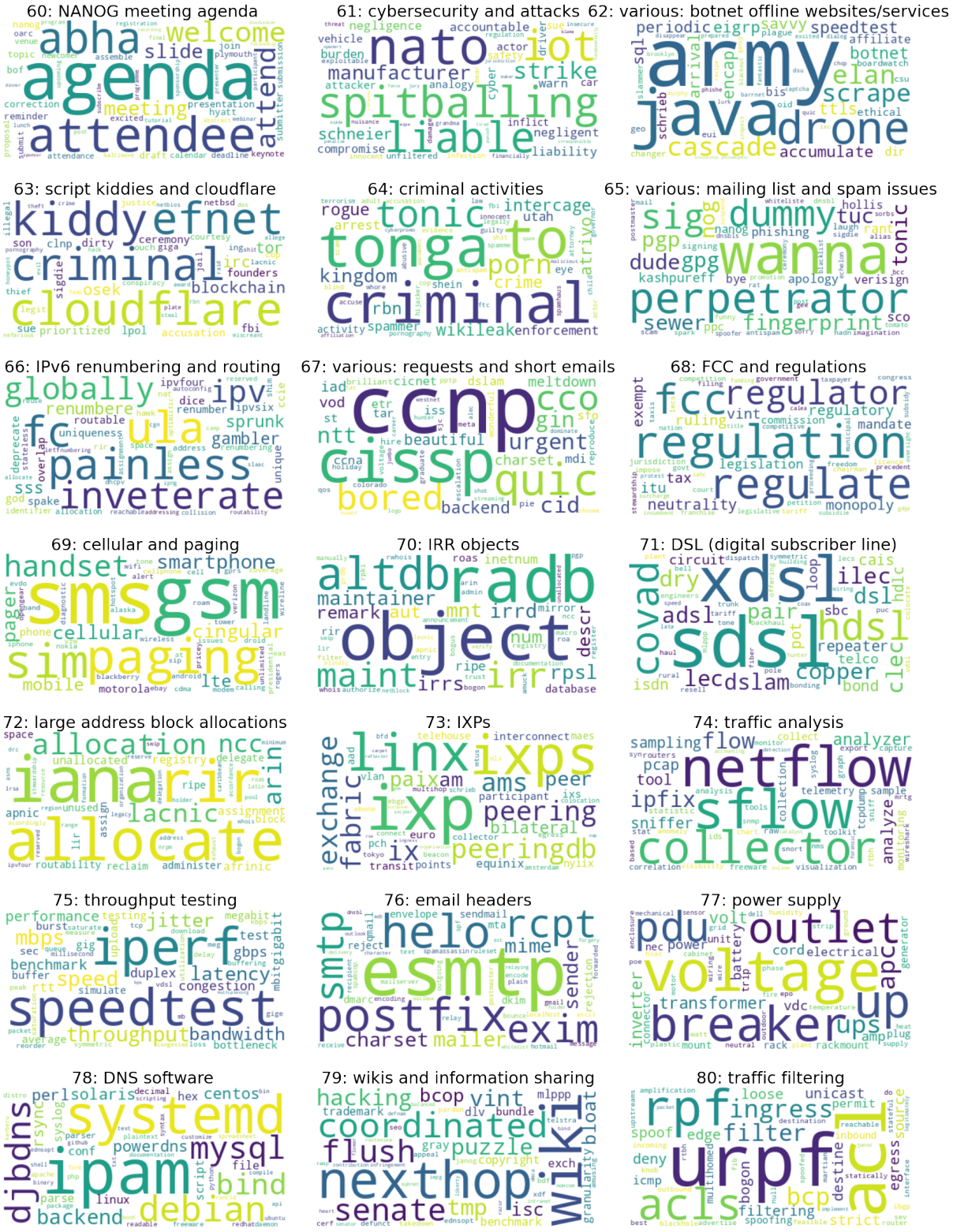


Figure B.4: Wordcloud representation of topic numbers 60 to 80.

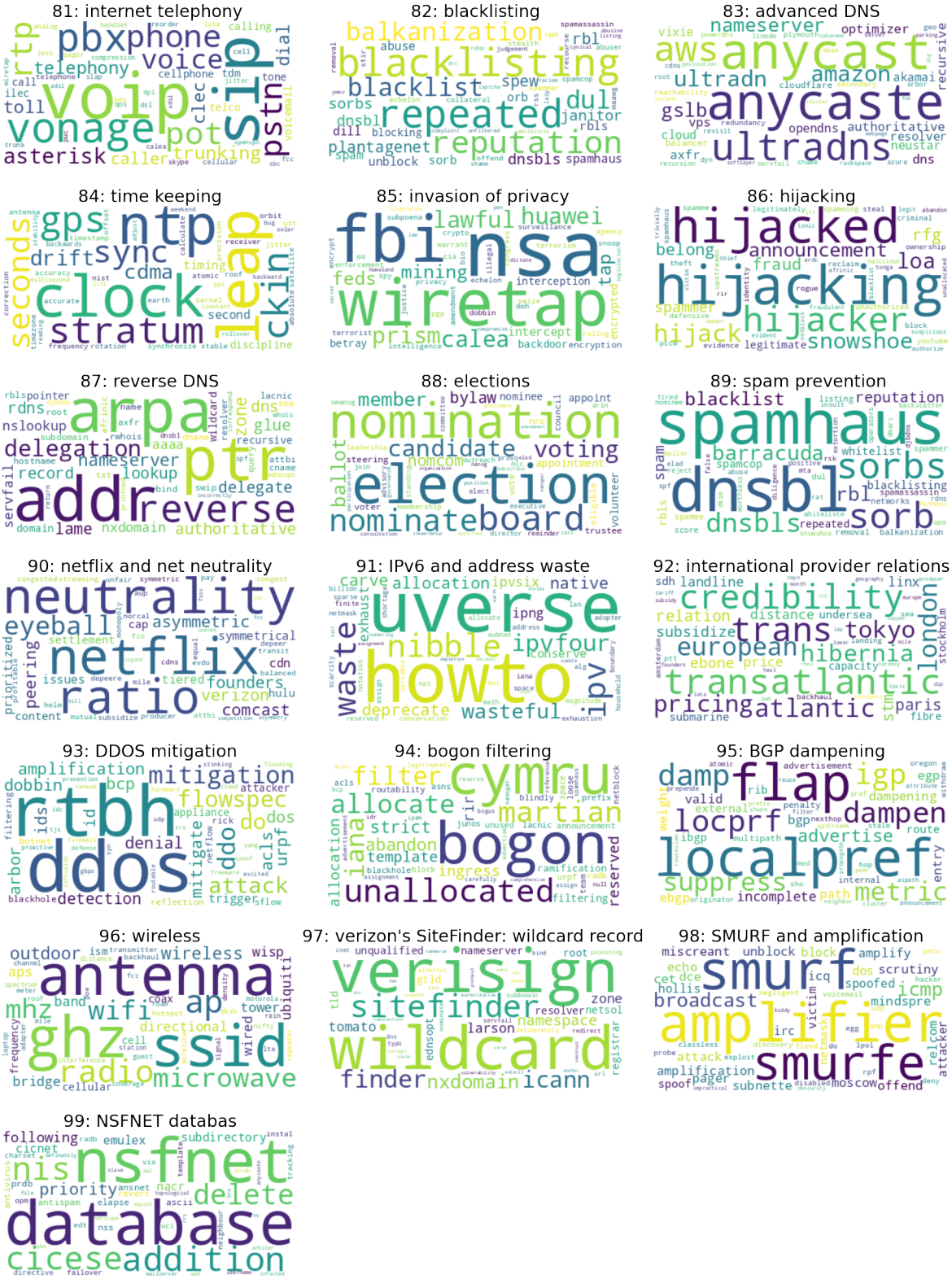


Figure B.5: Wordcloud representation of topic numbers 81 to 99.

Appendix C

Super Topics

1. "routing and protocols": "Internet reports," "DDOS attacks," "hardware specs," "BGP routing, metrics and announcements," "hardware models and brands," "networking skills and mentoring," "monitoring," "scripting," "firewalls," "intradomain routing," "data layer protocols," "vulnerabilities," "layer 2 hardware," "BGP monitoring," "MTU and fragmentation," "routing software," "special routing protocols: LDP, MPLS, SR," "IRR objects," "traffic analysis," "DNS software," "wikis and information sharing," "traffic filtering," "time keeping," "DDOS mitigation," "bogon filtering," "BGP dampening," "SMURF and amplification"

2. "spam and criminal activity": "email issues," "vulnerabilities and infections," "mailing list and subscriptions," "legislation and law enforcement," "SMTP," "cybersecurity and terrorism," "mixed thread talk," "cybersecurity and attacks," "various issues: botnet, websites, disappearing services," "sript kiddies and cloudflare," "criminal activities," "various: some mailing list and spam issues," "various requests and short emails," "email headers," "blacklisting," "invasion of privacy," "hijacking," "spam prevention"

3. "outages and providers": "outages," "Internet provider pricing models and regulation," "IXPs (mostly MAE-west)," "New York datacenters and bandwidth," "fiber cuts," "Cogent peering dispute," "power blackouts," "ISPs and peering," "international providers and groups," "FCC and regulations," "netflix and net neutrality," "international provider relations," "nsfnet database"

4. "address space": "IP dual stack," "multihoming," "DHCPv6 and SLAAC," "address space allocation," "LRSA (Legacy Registration Services Agreement," "subnet allocation," "routing tables and deaggregation," "IPv6 renumbering and routing," "large address block allocations," "IPv6 and address waste"

5. "server rooms": "server room and data centers," "fiber optic communication," "connect to hardware," "DSL (digital subscriber line)," "power supply"

6. "offlist contact": "offlist contact," "traceroutes," "centurylink issues," "geolocation and DNS issues"

7. "DNS": "domain name registrars," "DNS queries," "top level domains," "DNS," "certificates," "advanced DNS," "reverse DNS," "verizon's sitefinder: wildcard record"

8. "bandwidth and congestion": "congestion," "streaming," "CDN and P2P," "cellular and paging," "throughput testing," "internet telephony," "wireless"

9. "network operator group": "measurements and surveys," "NANOG meeting logistics," "call for presentations," "routing table reports," "IETG," "NANOG meeting agenda," "elections"