

Friedrich-Schiller-Universität Jena  
Institute of Computer Science  
Degree Programme Computer Science, B.Sc.

# Building a Mastodon Corpus for Conducting IR and NLP Experiments

## Bachelor's Thesis

Maximilian Ernst

1. Referee: Prof. Dr. Matthias Hagen
2. Referee: Jan Heinrich Merker

Submission date: November 14, 2024

## **Zusammenfassung**

Die Fachgebiete des Mikroblog- und föderierten Information Retrievals wurden in den vergangenen Jahren jeweils eingehend erforscht. Während föderiertes Information Retrieval das Suchen aus mehreren Quellen untersucht, die längere Dokumente enthalten, fokussiert sich Mikroblog-Retrieval auf von Nutzenden erstellte Kurzbeiträge, sogenannte Mikroblogs. Mikroblog-Daten von Twitter (inzwischen X) wurden bisher gewiss am häufigsten untersucht, jedoch hat in jüngerer Vergangenheit Mastodon, ein soziales Netzwerk basierend auf dem ActivityPub-Protokoll, verstärkt Aufmerksamkeit als eine Twitter-Alternative erhalten. Mastodon unterscheidet sich von zentralisierten Mikroblogplattformen durch seinen föderierten Aufbau. Die Schnittmenge von Mikroblog- und föderierter Suche ist bisher nicht erforscht, unter anderem da kein größerer passender Korpus entsprechender Dokumente existiert. In dieser Arbeit präsentieren wir solch einen Korpus, bestehend aus 3,6 Milliarden öffentlichen Mastodon-Posts, die innerhalb von 301 Tagen von 1.081 Instanzen gesammelt wurden.

Wir analysieren den Korpus hinsichtlich verschiedener Metriken, visualisieren die Ergebnisse und führen hinsichtlich des Information Retrievals ein Basisexperiment durch, welches das Potenzial des Korpus zu präsentieren sucht. Dieses Experiment hat die Erkenntnis gebracht, dass eine größere Dokumentenmenge zu durchsuchen nicht immer auch mehr relevante Ergebnisse liefert. Diese Arbeit strebt danach, der Startpunkt für das neue Fachgebiet der föderierten Mikroblogsuche zu sein. Wir werden Experimente mit dem Korpus via TIREx ermöglichen, um die Forschung voranzutreiben, ohne dabei jemandes Privatsphäre zu verletzen. Den Quellcode unseres rücksichtsvollen, parallelisierten Crawlers veröffentlichen wir zusammen mit dieser Arbeit.

## Abstract

The fields of microblog and federated information retrieval each have received decent attention in the past years. While federated retrieval studies settings with multiple resources containing longer documents, microblog retrieval focuses on short, user-generated content. In microblog retrieval, data from Twitter (now X) has surely been studied the most, but recently, a social network based on the ActivityPub protocol—Mastodon—has received increased attention as an alternative to Twitter. Mastodon differs from centralized microblogging platforms in that it is a federated network. The intersection of federated and microblog search has yet to be explored, but a larger suitable collection of documents did not exist. In this work, we present a corpus consisting of 3.6 billion public Mastodon statuses, crawled from 1,081 instances over a period of 301 days. We analyze the corpus with respect to various metrics, visualize our findings, and conduct a baseline information retrieval experiment with the corpus to preview its potential. In this experiment, we have found that searching a larger set of Mastodon statuses does not necessarily yield more relevant results. This work is intended to be a starting point for the new field of federated microblog search. We will host the corpus on TIREx to enable research without compromising user privacy, and we release the source code of our polite, parallelized crawler along with this work.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Related Work</b>	<b>3</b>
<b>3</b>	<b>Creating a Mastodon Corpus</b>	<b>6</b>
3.1	Crawler Software . . . . .	6
3.2	Instance Sample . . . . .	12
3.3	Crawling Instances . . . . .	16
3.4	Corpus Analysis . . . . .	17
<b>4</b>	<b>Conducting an IR Experiment using the Corpus</b>	<b>23</b>
4.1	Methodology . . . . .	23
4.2	Results . . . . .	25
<b>5</b>	<b>Discussion</b>	<b>27</b>
<b>6</b>	<b>Conclusion</b>	<b>30</b>

# Chapter 1

## Introduction

Mastodon, a federated free and open source social network, has become increasingly popular, especially in the past two years. After Twitter was acquired by Elon Musk and renamed X, Mastodon grew from about 0.4 to 1 million active users,<sup>1</sup> a 150% increase. As Mastodon offers microblogging features similar to those of the platform X, thousands of users have migrated to Mastodon since the acquisition [He+23]. Because Mastodon was only founded recently<sup>2</sup> and still has considerably fewer users, there has been less research using Mastodon data compared to data from X, which in contrast “has been one of the most popular social network sites for academic research”, as Yu and Muñoz-Justicia [YM20] have put it.

Compared to the social network X, Mastodon’s federated nature poses significant differences in how information is distributed. On centralized platforms like X, all data is in the hand of a single entity. A search on such a platform usually targets only a single resource which spans all user posts. When searching on Mastodon,<sup>3</sup> on the other hand, it is only possible to search a single one of the currently ca. 10,000 instances<sup>1</sup>. Searching a single Mastodon instance is information retrieval with the challenges of a microblog search, just like a search on X.

As part of the *fediverse*—federated universe, a network of social networks—Mastodon utilizes the *ActivityPub* protocol [Lem+18], an open W3C standard, for client- and instance-to-instance communication. Though instances do fetch information from one another, Mastodon is not designed to share all data between instances. Even if technically possible, it is highly unlikely that a

---

<sup>1</sup>According to <https://mastodon-analytics.com/>

<sup>2</sup>Initial commit on Feb 2nd, 2016:

<https://github.com/mastodon/mastodon/commit/9c4856b>

<sup>3</sup>Searching text in statuses is possible on all instances using a Mastodon version higher or equal to 4.2.0. See: <https://github.com/mastodon/mastodon/releases/tag/v4.2.0>

single instance will know every status (also called toot or post) of the network. Hence, unlike on the platform X, information retrieval on the whole Mastodon network additionally comes with the problems of federated search.

As a result, to find the most relevant documents to an information need in a network-wide Mastodon search, the problems of both federated and microblog search apply. The key problems of microblog retrieval like the sparsity inherent to very short documents and the difficult assessment of the quality [Nav+11] have been studied extensively in the TREC Microblog Search tracks in 2011–2015 [Oun+11; Sob+12; LE13; Lin+14; Lin+15]. Similarly, federated retrieval has been investigated, e.g., in the TREC 2013 and 2014 Federated Web Search tracks [Dem+13; Dem+14]. With the recently increased popularity of Mastodon, the importance of exploring the combination of federated search and microblog search has increased, too. We call this new research direction at the intersection of the two previously separately researched fields *federated microblog search*.

This thesis aims to promote the field of federated microblog search by building a large corpus of 3.6 billion Mastodon statuses from 1,081 instances spanning a time period of 301 days. The corpus will be hosted on TIREx, The Information Retrieval Experiment Platform [Frö+23a], which is built upon TIRA [Frö+23b], to enable privacy-preserving experiments in information retrieval (IR) and natural language processing (NLP). We analyze the resulting corpus regarding its size, languages, the level of federation, and other metrics. Alongside this work, we release our polite crawler used to create the corpus.<sup>4</sup>

To preview the potential of our corpus, we conduct a baseline retrieval experiment. We prepare 25 search topics consisting of title, description, and narrative derived from popular keywords in Google<sup>5</sup> and Twitter trends<sup>6</sup>, and most used hashtags from our corpus. Five retrieval systems with access to different sets of statuses from the corpus retrieve documents, and we partly judge their relevance. To fill the holes in the judgments, we utilize *autoqrels* [MS23], and we evaluate the effectiveness of our retrieval systems as well as the correlation between manual and automated judgments.

We have already presented the corpus in an earlier stage and the crawler at the First International Workshop on Open Web Search (WOWS) [Wie+24]. The repository containing this work’s software package and results can be found on Webis GitLab: <https://git.webis.de/code-teaching/theses/thesis-ernst>. All steps necessary to reproduce the results are described in that code repository.

---

<sup>4</sup><https://github.com/webis-de/mastodon-search>

<sup>5</sup><https://trends.google.com/trends/>

<sup>6</sup><https://archive.twitter-trending.com>

# Chapter 2

## Related Work

Our work on Mastodon as a microblogging social network is related to previous work on other microblogging services like the platform X<sup>1</sup>. As shown by Yu and Muñoz-Justicia [YM20], X has been very popular among academic research. Mastodon has received some attention, too, but not nearly as much. Papers about Mastodon focus, for example, on social relations and the structure of Mastodon and the fediverse [CGT21; CGT22; ZGR18; ZLG20], moderation [Ana+23; Bon+24], or investigate migration patterns between Mastodon and X [He+23; Jeo+24; WKR24]. Raman, Joglekar, Cristofaro, Sastry, and Tyson [Ram+19] have also pointed out a degree of centralization of Mastodon instances. Centralization impacts information retrieval on a Mastodon corpus, because the resource selection in federated search might tend to prefer large instances and ignore small ones, so this aspect should be considered.

Building a corpus of statuses from Mastodon is related to previously built microblog corpora. There are a number of corpora of the platform X, like the one used for the TREC Microblog Search tracks 2011 and 2012 [McC+12] or another corpus for event detection [MMJ13]. While they are useful for studying microblog search, they are unsuited for federated search. Mastodon corpora exist, but do not enable federated microblog search, since they have a different focus. E.g., Cerisara, Jafaritazehjani, Oluokun, and Le [Cer+18] crawled English statuses from a single instance and Álvarez-Crespo and Castro [ÁC24] created a Galician corpus from another single instance.

As a community effort, tools to search Mastodon (`instances.social`) or the fediverse (`fediversesearch.com`) exist. They target instances and thus resemble a manual resource selection, but do not index documents (i.e., statuses). Information retrieval on Mastodon has been done scholarly [TCH18], but does not easily enable subsequent experiments using the same dataset—for a reason. Dataset handling is a very delicate topic. Some community projects

---

<sup>1</sup><https://x.com/>

like FediMapper<sup>2</sup> which allowed accessing data via an API have been shut down because of user backlash. A paper that released a Mastodon dataset has been retracted [Zig+22], mentioning GDPR violations. Therefore, it is imperative for us to always respect user privacy and, foremost, never publish raw data from the corpus.

We build upon significant prior work in the fields of microblog search and federated search. Regarding the latter, Shokouhi and Si [SS11] postulated the three tasks of collection representation, collection selection, and result merging, where collection is a synonym for resource. Resource representation means a broker needs to store data about resources of which the contents are not entirely known to the broker. We can ignore this task in our setting, since we collect all statuses ourselves and thus know all contents of every resource (i.e., Mastodon instance). The other two tasks have also been the tasks of the TREC Federated Web Search track in 2013 [Dem+13]. Resource selection, the challenge to pick those resources which contain the most relevant documents to an information need (i.e., topic), appears to be the most challenging task in our context. Contrary to the TREC scenario, our resources are almost uniform in terms of outer document definitions, but the number of resources is much larger. For this task, the centralization of the network has to be considered, but also the limited capabilities of small instances. Result merging is the task of combining the results from querying the different resources. Because of the homogeneity of all retrieved documents and the fact that they actually are microblog documents, the result merging task can be altered to a microblog search in our setting. The TREC Federated Web Search track in 2014 [Dem+14] added the task of vertical selection where queries are classified into content dedicated to categories like news, finance, or image. Although this task might become relevant in the future, it is out of scope for this work. For now we focus on the task of resource selection.

Regarding microblog search we most notably build upon the TREC Microblog Search tracks in the years 2011–2015 [Oun+11; Sob+12; LE13; Lin+14; Lin+15]. The tracks of 2011–2013 set the task of ad hoc search, where at a specific time  $t$  the most relevant documents to a topic shall be retrieved. In 2013, the TREC Microblog Search track introduced evaluation as a service, an effort to prevent issues with the redistribution of real-world datasets. Tweets were instead retrievable via an API. TIRA [Frö+23b], TIRA Integrated Research Architecture, takes this even further and enables experiments by running containerized software submissions itself. Submitters merely receive the results of their experiment, but not directly data from the corpus. In 2014, a timeline generation task was introduced, which was called email digest in 2015,

---

<sup>2</sup><https://github.com/tedivm/fedimapper>



and is the challenge of summarizing the content of documents relevant to a topic. Searching one Mastodon instance resembles the ad hoc search task and is our focus in this work, though result summarization might be explored in the future.

As overviewed by Kumar and Sharma [KS18], there are a number of variables to consider in information retrieval, mainly the architecture of the retrieval system and the retrieval model. Elasticsearch tackles some of these challenges like document storage and indexing, search functionality, and a relevance measure which is the well-established Okapi BM25 [Rob+94] by default. Retrieval experiments are commonly conducted and evaluated following the Cranfield paradigm. One of the assumptions of this paradigm—all relevant documents to a topic are known— [Voo01] is often violated because of its unfeasibility. As a consequence, holes in the relevance judgments of retrieved documents might occur. Efforts to deal with such holes, like the bpref measure [BV04] or precision estimation [YA06], are not always adopted, but a number of ways to evaluate effectiveness even without human relevance judgments already exist [Roi+20]. Recent papers have also shown good results utilizing LLMs to patch holes in relevance judgments [UKL24]. The *autoqrels* tool<sup>3</sup> from one such paper by MacAvaney and Soldaini [MS23] is used for the evaluation of our information retrieval experiment. While explicitly stating that this tool is not able to always judge the relevance of single documents accurately, the authors achieved good results in ranking systems.

---

<sup>3</sup><https://github.com/seanmacavaney/autoqrels>

# Chapter 3

## Creating a Mastodon Corpus

To build our Mastodon corpus, we identified three necessary main steps: (1) writing crawler software to retrieve statuses from Mastodon, (2) deciding how many and which instances to crawl, and (3) crawling these instances. After explaining our approach to these three steps, we analyze the resulting corpus.

### 3.1 Crawler Software

**Crawler Architecture** As a very first step, we scanned the Web for existing software to build upon. While there are a few projects which point in a similar direction,<sup>123</sup> none fit our needs closely enough or were well-documented in a way that it was worth the investment of time to read, understand, and utilize them. We therefore decided to develop our own crawler.

Because we are most knowledgeable about it and for the abundance of packages it offers, we settled for Python as programming language. The software is written as a Python package with a few different commands. For crawling, the command is `stream-to-es` which retrieves statuses from a single Mastodon instance. This command relies heavily on Mastodon’s REST API, specifically the timelines API<sup>4</sup>, and *Mastodon.py*’s<sup>5</sup> `Mastodon` object to interact with the API. Command line parameters of this command are a URL of an Elasticsearch installation, credentials of an Elasticsearch user, and a domain of a Mastodon instance. Until manually aborted or an error is encountered, statuses from the specified Mastodon instance are retrieved and saved to Elasticsearch by the program.

---

<sup>1</sup><https://github.com/tedivm/fedimapper>

<sup>2</sup><https://github.com/MarcTOK/Franck>

<sup>3</sup>[https://github.com/alexdrk14/Mastodon\\_crawler](https://github.com/alexdrk14/Mastodon_crawler)

<sup>4</sup><https://docs.joinmastodon.org/methods/timelines/#public>

<sup>5</sup><https://github.com/halcy/Mastodon.py>

Initially, `stream-to-es` connects to the given Elasticsearch installation trying to find statuses of the Mastodon instance from a previous crawl. If successful, the `id`<sup>6</sup> of the last crawled status is retrieved. An intermediate crawl is performed next: If the crawler did not receive an `id` from Elasticsearch, it requests the timelines API once to fetch the latest 40 statuses. If Elasticsearch yielded an `id`, the `id` is passed as `min_id` parameter to the API. The instance then returns statuses with `ids` immediately newer than `min_id`, effectively acting like a bookmark. We repeatedly use the `id` of the latest of the newly fetched statuses as `min_id` parameter for the next API request to paginate through statuses without gaps. As the crawler also always sends the argument `limit=40` to the API to receive the maximum number of statuses, it stops the intermediate crawl after receiving less than 40 statuses, because there are no newer statuses in this case.

After catching up on missed statuses, the crawler tries to connect to the Mastodon instance's streaming API<sup>7</sup>: An HTTP connection is held open via which the instance immediately pushes newly received statuses to all streaming clients, without the need to poll. The crawler will try to reconnect to the streaming API on connection loss. Only after 5 consecutive unproductive streaming connections will the crawler give up streaming and fall back to polling mode.

The majority of instances disallows streaming categorically. In this case, the crawler falls back to polling mode immediately. This mode repeatedly requests the timelines API with the `min_id` parameter to fetch new statuses, as explained above, but without stopping. The waiting time between requests is dynamically adapted based on the number of statuses received per request: On receiving 40 statuses, the crawler reduces the waiting time between requests, because likely more statuses reach the timeline than we are retrieving, but never below one second. On receiving 10 or less statuses, waiting time is prolonged, but never above 60 minutes. The waiting time between requests increases faster the less statuses are received. With this adaptive behavior we try to reduce stress for servers to an absolute minimum.

The crawler also uses the `Session` object of the *Requests* package<sup>8</sup>. It is passed as an argument to the `Mastodon` object and serves multiple purposes: The `Session` object accepts an instance of the `LimiterAdapter` object from the package *Requests-Ratelimiter*<sup>9</sup> as an argument which allows to limit the number of requests to the server. We use it as another safeguard to not strain an instance with more than one request per second. One of the possible

---

<sup>6</sup><https://docs.joinmastodon.org/entities/Status/#id>

<sup>7</sup><https://docs.joinmastodon.org/methods/streaming/#public>

<sup>8</sup><https://github.com/psf/requests>

<sup>9</sup><https://github.com/JWCook/requests-ratelimiter>

parameters for `LimiterAdapter` is, in turn, a `Retry` object from the *urllib3* package<sup>10</sup>. We use it to resend failed requests after errors with a waiting time between each consecutive retry double the one before, starting at one second after the second try. With a maximum of 14 retries per error category, the crawler keeps running even if the instance is down for a few hours. The `Session` object instance also adds our user agent<sup>11</sup> to requests, which identifies us to instance admins, offering them a way to contact us. One more way we utilize the `Session` object is by trying to reuse TCP connections—another measure towards using as little of an instance’s resources as possible.

Regarding users, i.e., the authors of statuses, we strive to respect their privacy settings in our crawl. There is a Mastodon account setting which the API calls `noindex`, and on Mastodon is described as follows: “Include public posts in search results”. This setting is evaluated for every status at crawl time. When the author does not want to be included (i.e., `noindex` is true), only the current time, the status’s `id`, and the `noindex` setting itself are stored for this status.

While the primary thread of the crawler is responsible for crawling an instance, it also starts two other threads. One of these is printing status updates to standard output after the first minute of running and then every ten minutes, which tells the user whether the crawler is still running and fetching new statuses. The other secondary thread is storing the statuses to Elasticsearch, emptying the crawler’s double-ended queue (deque) of statuses, as well as keeping track of the time passed since last emptying the deque, as explained in the following section. Because both secondary threads only execute very little Python bytecode, the crawler is not noticeably affected by the global interpreter lock<sup>12</sup>.

**Storing Statuses to Elasticsearch** Received statuses are prepared for Elasticsearch storage by using the package *Elasticsearch DSL*<sup>13</sup>. Inheriting from this package’s `Document` class, we largely replicated Mastodon statuses, mapping most of a status’s data fields (details in 3.1, Data Fields of Stored Statuses) to Elasticsearch counterparts in a `Status` object. Elasticsearch DSL allows to transform instances of `Document` to a dictionary. We implemented the FIFO principle with a deque, to which the status dictionaries are input and the output is an Elasticsearch instance.

The crawler stores all received statuses to given Elasticsearch instance—a

---

<sup>10</sup><https://github.com/urllib3/urllib3>

<sup>11</sup>Webis Mastodon crawler (<https://webis.de/>, [webis@listserv.uni-weimar.de](mailto:webis@listserv.uni-weimar.de))

<sup>12</sup><https://wiki.python.org/moin/GlobalInterpreterLock>

<sup>13</sup><https://elasticsearch-dsl.readthedocs.io>

130 nodes cluster hosted by the Webis group<sup>14</sup> in our case. In Elasticsearch, every status has mandatory metadata attached to it. Elasticsearch will generate this metadata when storing a status, or the metadata can be specified in advance. Part of the metadata is an ID which uniquely identifies each status. If a status with an already existing Elasticsearch meta ID is stored, the status is overwritten. Using Python’s *uuid* package, the crawler calculates a UUID version 5 which always produces the same output from an input. With the crawled instance’s domain and the status `id` as input, the resulting UUID is passed to Elasticsearch as the status’s meta ID. Thus, we store the same status multiple times when crawled from different instances, but only once from each instance, even if fetched repeatedly.

To limit requests to the Elasticsearch installation, we utilized the *Elasticsearch*<sup>15</sup> package’s `streaming_bulk` method. Multiple items can be grouped together in chunks and committed to Elasticsearch at once with this method. Once a minute the crawler checks if it holds at least a full chunk of 500 statuses. If this condition is met or at least 30 minutes have passed, all statuses are committed in chunks to the Elasticsearch instance until the status deque is empty, then the timer is reset.

Due to the number of instances we crawled and hence the large amount of information to store, we split data into a separate Elasticsearch index per month by appending `_YEAR_MONTH` to the fixed base index name. An index template<sup>16</sup> ensures every index of the crawl always has 2 replicas and 20 shards, to not overload the Elasticsearch instance.

**Data Fields of Stored Statuses** We tried to replicate Mastodon statuses as close as possible in our crawl. We adhered to the Mastodon naming scheme for every data field which we stored verbatim to make it easy to understand our data. Thus, the Mastodon documentation<sup>17</sup> also acts as a reference for the description of our corpus’s field names. A few fields which were unnecessary to us were dropped by our crawler prior to storing. Examples of such fields are `like` and `boost` (also called `reblog`) counts as they are always zero when a status first appears on a timeline, i.e., when we fetch it. Boosts originally include the entire post that was boosted, including account data. In contrast, we only store `id` and `url` of the boosted status since the boosted status can be fetched when needed and to save some storage space.

Additionally, we included some fields that are not originally part of Masto-

---

<sup>14</sup><https://webis.de/>

<sup>15</sup><https://github.com/elastic/elasticsearch-py>

<sup>16</sup><https://www.elastic.co/guide/en/elasticsearch/reference/current/index-templates.html>

<sup>17</sup><https://docs.joinmastodon.org/entities/Status/>

don statuses. Purpose of these fields is to make it easier to create corpus statistics or compare statuses, and to make corpus utilization more comfortable. For example, there is no field to easily and uniquely identify a user.<sup>18</sup> Our custom field `handle` does exactly that—it is always `<USERNAME>@<INSTANCE>`. We also added the time at which the status was received (`crawled_at`), the instance we crawled to receive the status (`crawled_from_instance`), the instance on which the status was published (`instance`), a boolean indicating whether the status’s author is registered at the crawled instance (`is_local`), and a few other fields to every status. Notable fields we stored, their data type, and whether they are always present are listed in Table 3.1. Again, refer to the Mastodon documentation for explanations on what information is stored in each field.

**Table 3.1** Fields of a status as represented in Elasticsearch. Mandatory fields will always have a non-`null` value (but might be a zero-length string). Fields with sub-fields are mere containers for other fields. Because the value of `object` type fields can be either `null` or a single object, their sub-field count is an upper bound. `List` type objects can hold any amount of object that is a non-negative integer, hence:  $j, k, l, m, n, o \in \mathbf{N}$ . `List` or `object` types themselves are not included in the sub-field counts, only their sub-fields of other types.

Field	Mandatory	Type	Sub-fields
<i>Fields of a Status (top-level)</i>			
<code>content</code>	✓	string	
<code>crawled_at</code>	✓	date	
<code>crawled_from_instance</code>	✓	string	
<code>created_at</code>	✓	date	
<code>id</code>	✓	string	
<code>in_reply_to_id</code>	✗	string	
<code>instance</code>	✓	string	
<code>is_local</code>	✓	boolean	
<code>language</code>	✗	string	
<code>sensitive</code>	✓	boolean	
<code>spoiler_text</code>	✗	string	
<code>uri</code>	✓	string	

Continued on next page

<sup>18</sup>E.g., the field `acct` is `<USERNAME>@<INSTANCE>` for remote accounts and just `<USERNAME>` for accounts local to an instance.

Table 3.1 (Continued)

Field	Mandatory	Type	Sub-fields
visibility	✓	string	
application	✗	object	2
emojis	✗	list	$4j$
mentions	✗	list	$4k$
poll	✗	object	$6 + 2l$
reblog	✗	object	2
tags	✗	list	$2m$
5 others		various	
<i>Sub-fields of account, describing an author</i>			
bot	✓	boolean	
created_at	✓	date	
discoverable	✗	boolean	
display_name	✗	string	
group	✓	boolean	
handle	✓	string	
id	✓	string	
noindex	✗	boolean	
url	✓	string	
15 others		various	$4n + 3o$
<i>Sub-fields of card, describing a website preview</i>			
description	✗	string	
image	✗	string	
language	✗	string	
title	✗	string	
url	✗	string	
11 others		various	
<i>Sub-fields of media_attachments, i.e., image, video, or sound files</i>			
description	✗	string	
url	✗	string	
5 others		various	17

## 3.2 Instance Sample

**Gathering Data about Instances** With the crawler software ready, we needed to choose how many and which Mastodon instances to crawl. Naturally, the first step towards there is to find every available instance. *Minoru's Fediverse crawler*<sup>19</sup> makes a good effort to find all fediverse instances by looping through the steps of checking if the instances already known are alive, fetching their peers<sup>20</sup> list, and compiling a summary list of alive instances. We use the list provided by the developer's own installation of this software,<sup>21</sup> downloaded on 18th December 2023, which contains 22,178 domains.

**Table 3.2** Deduplicated number of instances we were able to fetch activity and NodeInfo data from, after runs of our `obtain-instance-data` command.

Run	Instances
1	9,595
2	10,086
3	10,339
4	10,346
5	10,346
6	10,347
7	10,351

This list of instances consists of domains running various fediverse software, not just Mastodon. To get data about the software used on every domain, we added another command to our software called `obtain-instance-data`, which can be run on different kinds of input. One input option is a single-line JSON array from a file, like the aforementioned list of domains. The program starts one thread to write all gathered data to a user-specified output file in JSONL format. Then, one thread per domain from the input file is run. Six to eight concurrent threads seems to be a good tradeoff between speeding up the process and not exceeding the rate limit for new connections by internet service providers. These per-domain threads use `Mastodon.py` to fetch activity<sup>22</sup> and NodeInfo<sup>23</sup> data from an instance. It should be noted that receiving an answer to one of these two requests does not mean an answer to the other request is guaranteed. After the last per-domain thread finishes, the program exits.

---

<sup>19</sup><https://github.com/Minoru/minoru-fediverse-crawler>

<sup>20</sup>A peer is a domain that an instance is aware of, according to the Mastodon documentation. See: <https://docs.joinmastodon.org/methods/instance/#peers>

<sup>21</sup><https://nodes.fediverse.party/nodes.json>

<sup>22</sup><https://docs.joinmastodon.org/methods/instance/#activity>

<sup>23</sup><https://github.com/jhass/nodeinfo>



As a lot of gaps in the acquired data are to be expected, the command's second mode of operation tries to fill in some of these gaps. The program reads its own output file from a previous run and executes the same steps as explained above, except it skips requests where data is already present. Even with a request timeout of 30 seconds, running the command multiple times this way notably increases the amount of data and thus the instances to work with, as shown in Table 3.2. We ran the command a total of seven times. Instances we were unable to get activity and NodeInfo data from are mostly not running Mastodon or, if they are, they are either unresponsive or in some way access restricted. Both these categories of instances are unsuitable for our crawl and were sorted out. The final, deduplicated number of instances which we have both activity and NodeInfo data for is 10,351, all running Mastodon or compatible (i.e., forks of Mastodon) software. These are the candidate instances for sampling.

**Sampling Instances** After finding the candidate instances, we needed to decide on a sample size. Crawling more instances would increase the possibilities of the resulting corpus, but the capability of our cluster posed a limit in terms of disk space and how many instances we could crawl simultaneously. We settled on a sample size of 1,000 instances as a tradeoff between these two factors.

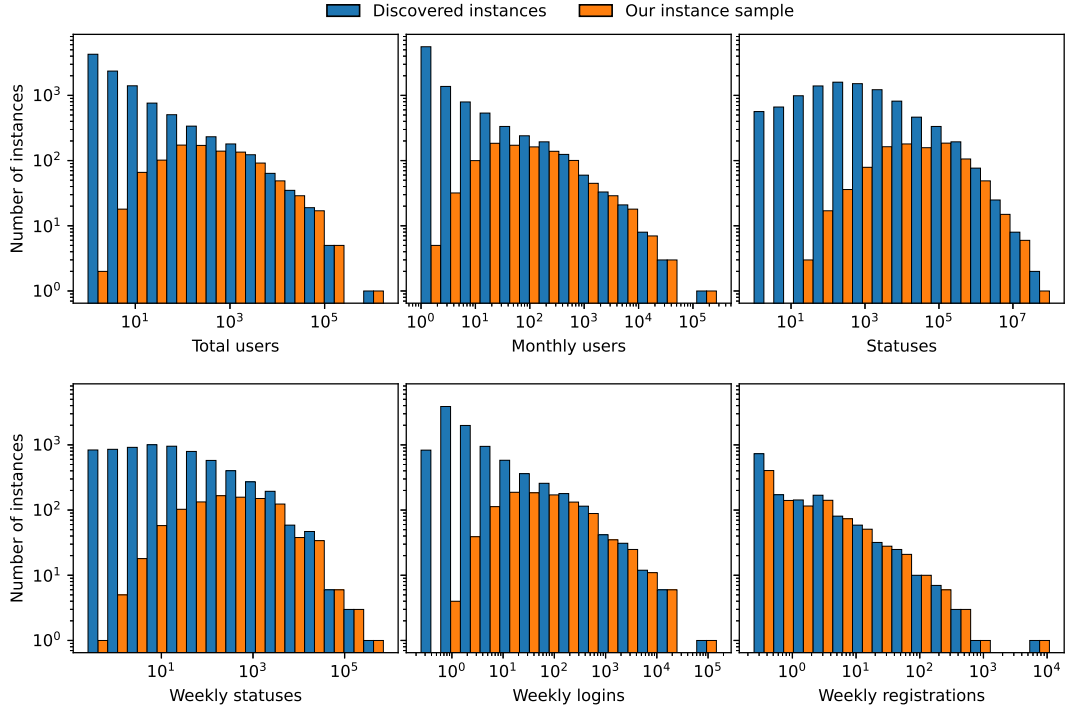
To find a proper way of sampling instances, we first analyzed the set of candidate instances regarding different metrics. Out of the metrics we had available from NodeInfo and activity data, we regarded the following six:

- Total users: number of users registered on the instance
- Monthly users: number of users who signed in at least once during the last 30 days
- Statuses: number of statuses by users registered on the instance
- Weekly statuses: number of statuses per week by users registered on the instance
- Weekly logins: number of logins per week on the instance
- Weekly registrations: number of registrations per week on the instance

Loading all activity data is done by our **Analyzer** object, which is another part of our Python package. When an instance of this object is created, a JSONL file output by the command for the previous step—`obtain--instance-data`—is read and the data is put in a single *pandas DataFrame*<sup>24</sup>.

---

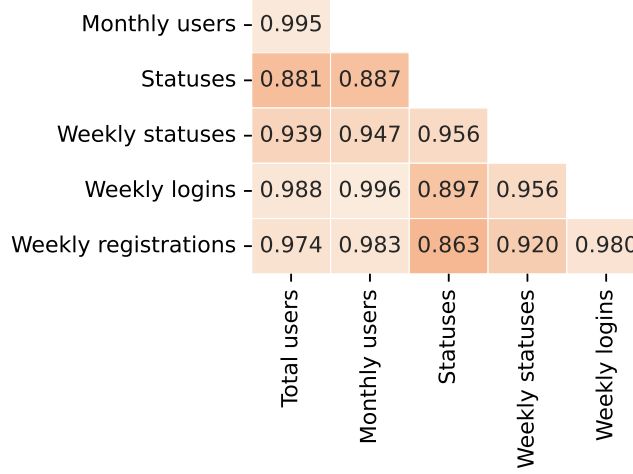
<sup>24</sup><https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html>



**Figure 3.1:** Histograms showing the distribution of instances regarding six activity metrics with a log scale on both axes. The entire set of the 10,351 candidate instances is colored blue, orange represents our sample set of 1,000 Mastodon instances. Number of statuses, and total and monthly user count are taken directly from the Mastodon API. Weekly metrics are the arithmetic mean over the four most recent full weeks from activity API data. Adapted from Wiegmann, Reimer, Ernst, Potthast, Hagen, and Stein [Wie+24].

The activity API provides the weekly amount of statuses, logins and registration for the last several weeks. Our weekly values are the arithmetic mean over the most recent four full weeks.

In the process of loading the data, obvious duplicates are filtered out. A duplicate in this context is another notation for the same effective domain. Two duplicates with/without trailing dots and three duplicates originating from internationalized domain names/punycode were filtered out. For the analysis only we also sorted out instances with implausible data. We considered a negative number of statuses implausible and sorted out one such instance. Another instance, which we removed for analysis, reported obviously untrue numbers: For one of its two total users it reported 97 billion (!) followers and 97 million statuses, which at the time of our analysis was more than all ca. 1.75 million users on the largest instance `mastodon.social` combined had authored.



**Figure 3.2:** Heatmap showing the Pearson coefficients for the pairwise correlation between the six activity metrics.

The distribution of instances over the aforementioned six metrics is shown in Figure 3.1 in blue color, and seems to be roughly log-normal with a skew to the left. The large amount of instances with lower user count and activity would lead to these instances dominating a completely random or stratified sample. Contrary, our goal for the sample was to include the majority of the most active instances, while still including enough of the lesser active instances, which is why we conducted weighted sampling [Wie+24]. We defined the six activity metrics to be random variables:  $X_1 := \text{total users}$ ;  $X_2 := \text{monthly users}$ ;  $X_3 := \text{statuses}$ ;  $X_4 := \text{weekly statuses}$ ;  $X_5 := \text{weekly logins}$ ;  $X_6 := \text{weekly registrations}$ . Although that is not necessarily correct—as can be seen in Figure 3.2—we assumed these variables to be pairwise independent for simpler calculations. We fit a probability density function (PDF) to each of the six metrics’ observed data. For every observed value we calculated the observation probability  $P(X_n); n \in \{1, 2, \dots, 6\}$  using the PDF, resulting in six probability values per instance  $i \in I$ . The sum of these six separate values describes the combined probability  $P(X_1, X_2, \dots, X_6)$ —given their assumed independence—which is the probability to observe an instance with the six activity values that were actually measured. We discounted each instance’s weight  $w_i$  by this combined probability to get the sampling probability:

$$P(i) \approx \frac{w_i}{\sum_{k \in I} w_k} \quad \text{with} \quad w = \frac{1}{P(X_1, X_2, \dots, X_6)}$$

Calculation and sampling is done by the command `choose-instances` of our Python software. The program also tries to fetch the timeline of every

sampled instance, sorts out all uncrawlable instances and replaces them with newly sampled instances, all in a loop until every instance of the sample is crawlable. Our final sample can be found in this work’s git repository.<sup>25</sup> As can be seen in Figure 3.1 in orange color, the distribution of sampled instances over the six activity metrics has less skew than the distribution of candidate instances, and matches our goal.

### 3.3 Crawling Instances

**Deployment** For deployment, we containerized the crawler using Docker. The steps undertaken in the image building process are defined in our Dockerfile.<sup>26</sup> Our image is based on the official Python image<sup>27</sup> with Python version 3.12. In the build process, our package’s `pyproject.toml` file is copied to the image. Python’s package manager `pip`<sup>28</sup> reads the dependencies listed in this file and installs the necessary packages inside the image. Finally, our own package is copied. We pushed the image resulting from the Docker build process to Webis’s registry where it is accessible for the Kubernetes cluster.

Creating a Helm chart<sup>29</sup> of our crawler and its image allowed us to deploy an arbitrary number of containers with a single command. The files defining the Helm chart can be found in the git repository<sup>30</sup> as well. They state which image to use, where to find the Elasticsearch installation, limits for CPU and memory usage, and how to handle the command line input. The latter should contain Elasticsearch credentials and a path to a file containing all Mastodon instance domains to crawl. With this setup, the command `helm install` deploys a container per instance to Kubernetes.

**Time Frame and Resampling** We started crawling the sampled instances on December 21, 2023 at 14:15 hrs. UTC+1. The crawl is currently still in process. We plan on keeping it running until we gathered statuses for a full year. With the amount of sampled instances, it was to be expected that some instances would become unavailable during the time of our crawl. Most of these instances were simply taken offline for reasons unknown to us. Other

---

<sup>25</sup><https://git.webis.de/code-teaching/theses/thesis-ernst/-/blob/main/data/instances.txt>

<sup>26</sup><https://git.webis.de/code-teaching/theses/thesis-ernst/-/blob/main/code/Dockerfile>

<sup>27</sup>[https://hub.docker.com/\\_/python](https://hub.docker.com/_/python)

<sup>28</sup><https://pip.pypa.io/>

<sup>29</sup><https://helm.sh/docs/topics/charts/>

<sup>30</sup><https://git.webis.de/code-teaching/theses/thesis-ernst/-/tree/main/code/helm>

reasons forcing us to stop crawling individual instances were timeline APIs set to nonpublic (requiring a token and thus a registration on the server), bot protection pages, and we seemingly got blocked by one instance. Instances were only excluded from crawling after several days of downtime. Based on the data from our initial sampling, we replaced each dropped instance with the one closest in sampling weight (see 3.2, Sampling Instances).<sup>31</sup>

When prior unavailable instances turned available again, we re-added them to the pool of candidate instances for possible resampling. As of October 17, 2024, we removed and resampled 82 unique domains. This number does not include dropped instances which were later resampled (e.g., due to extended downtimes). Since the crawler fetches missed statuses (see 3.1, Crawler Architecture), there are no gaps in the corpus for such an instance even if not crawled for months. For the same reason, reinstalling the entire Kubernetes deployment proved to have no effect on the resulting corpus, which allowed us to update to a new set of instances after resampling some of them.

## 3.4 Corpus Analysis

**General** The numbers regarding the corpus are still subject to change and thus mostly rounded or estimates. Analyzed is from October 17, 2024. We gathered a new set of activity and NodeInfo data, as described in 3.2, Gathering Data about Instances, mainly on the same day, for more accurate statistics. Cardinalities by Elasticsearch, which we used to get most of the numbers, are approximate values, based on the HyperLogLog++ algorithm.<sup>32</sup>

We have been crawling for 301 days from a total of 1,081 instances (but not from every instance all the time). 922 instances which were included in the initial sample are still being crawled today. The corpus contains 3.6 billion statuses, 174 million of which are unique. This equals a ratio of 0.05 of original posts to duplicates and means we crawled every status from an average of 20.5 different instances. We collected statuses from 24 thousand unique instances and 840 thousand unique users.

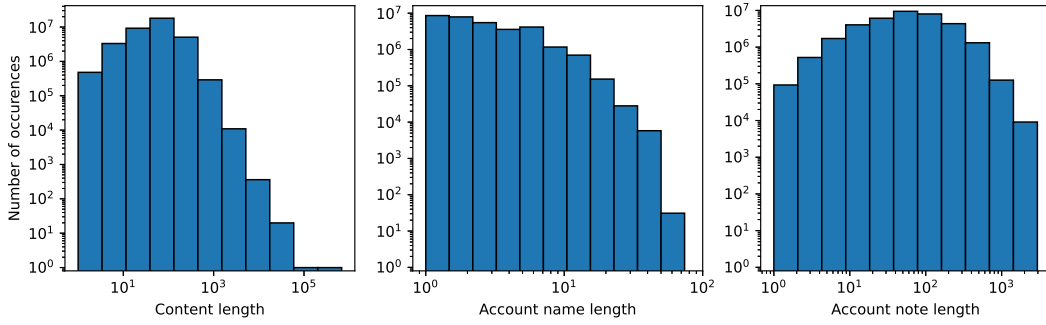
Elasticsearch can output the length of string fields, which it measures in token count.<sup>33</sup> A selection of relevant string fields and their length distribu-

---

<sup>31</sup>All changes and their reasons can be examined in the corresponding changelog: <https://git.webis.de/code-teaching/theses/thesis-ernst/-/blob/main/data/CHANGELOG.md>

<sup>32</sup>[https://www.elastic.co/guide/en/elasticsearch/reference/8.15/search-aggregations-metrics-cardinality-aggregation.html#\\_counts\\_are\\_approximate](https://www.elastic.co/guide/en/elasticsearch/reference/8.15/search-aggregations-metrics-cardinality-aggregation.html#_counts_are_approximate)

<sup>33</sup><https://elastic.co/guide/en/elasticsearch/reference/current/token-count.html>



**Figure 3.3:** Histograms showing the distribution of lengths of three different indexed string fields with a log scale on both axes. Length is measured in the default token count of Elasticsearch.

tions can be seen in Figure 3.3. As expected, the content length seems to be log-normally distributed (with a left skew), with the majority of statuses having a length of 100 tokens or less. A few statuses reach a length of over 100,000 tokens. Upon manual inspection, it seems to be some sort of trolling, since they contain a lot of mentions of the same account and an insult. Most of the account names are one to ten tokens long with a maximum length below 100. Account notes appear to be log-normally distributed, too, with almost no skew. The peak length is at ca. 50 tokens, overall ranging from one to a low four-digit number.

**Contributing Instances** Table 3.3 shows how many statuses were contributed to the corpus from crawling various instances or sets of instances and how many users were observed on them. We explicitly listed the 10 largest instances by the number of unique statuses crawled from their public federated timeline. Out of 174 million unique statuses in our corpus, we observed 78 million or 45 % on the timeline of `mastodon.social` alone. This number was smaller on other large instances, but still in the same order of magnitude. Similarly, 495 thousand or 59 % of all unique users could be observed on `mastodon.social`, with less on other explicitly listed instances but in the same order of magnitude. The size of `mastodon.social` compared to other instances is much more apparent when considering local timelines only, i.e., local statuses (posted by a user registered on the instance) crawled from the originating instance. The corpus contains 15.7 million such statuses of `mastodon.social`, over a third of all statuses crawled from local timelines. From the local timeline of `fedibird.com`, the second largest instance in this metric, we crawled 2.5 million statuses. Based on our corpus’s data, `mastodon.social` is also home to 185 thousand of the observed users, which is 28 % of users from local timelines, or 22 % of all users in the corpus. The instance with the second most

**Table 3.3** Number of statuses (= posts) and users per instance. Rows are each of the ten largest instances by crawled federated posts, all 1,071 other instances we crawled, all 1,081 crawled instances combined, all 8,464 instances from our initial sampling that we could still fetch NodeInfo data from, and all 9,520 instances from a current list of instances that we could fetch NodeInfo data from. Columns are unique statuses and users observed on an instance’s timeline—all (federated) or only considering users who are registered on the instance(s) only (local)—and number of statuses and users as per NodeInfo data.

Instance(s)	Crawled				NodeInfo	
	Federated		Local		Local	
	Posts	Users	Posts	Users	Posts	Users
mastodon.social	78.8M	495k	15,692k	185,028	107.5M	2,144,502
mastodon.online	48.7M	352k	1,249k	7,105	10.0M	190,285
fedibird.com	48.7M	230k	2,531k	6,631	19.6M	39,214
mstdn.social	46.8M	233k	1,598k	13,194	18.3M	241,034
mas.to	40.4M	214k	804k	9,986	9.4M	177,440
mastodon.world	39.4M	222k	774k	6,390	6.5M	184,958
ohai.social	38.0M	305k	85k	1,365	1.6M	39,544
social.vivaldi.net	37.2M	172k	535k	4,744	2.5M	55,993
universeodon.com	36.5M	321k	367k	3,307	3.7M	81,597
flipboard.social	35.9M	371k	176k	1,182	0.3M	4,771
1,071 others	156.0M	798k	19,470k	417,837	304.4M	2,850,350
1,081 crawled	174.4M	849k	43,281k	656,769	483.8M	6,009,688
8,464 initial					819.7M	7,031,949
9,520 current					864.2M	7,060,785

observed users on their local timelines, `mstdn.social`, only has 13 thousand users or 7 % the amount of the largest instance.

In NodeInfo data, the ratios of statuses and users between (sets of) instances are roughly similar to those of the local timelines from our corpus; the absolute numbers are about one order of magnitude higher. According to NodeInfo data, there are a total of 819.7 million statuses and 7 million users on the 8,464 instances from our initial analysis which are still reachable today. Thus, our corpus currently holds 21.3 % of the statuses and 12.1 % of the users on these instances of all time. 2,449 Mastodon instances became unavailable in the time of our crawl.<sup>34</sup> From a current list of instances, we were able to fetch NodeInfo data from 9,520 instances,<sup>35</sup> which means at least 1,056 instances went online (again) in the last 10 months. These additional instances would add 44.5 million or 5.4 % posts and 28.8 thousand or 0.4 % users.

**Source Instances** Because we crawled federated timelines, the corpus does not only consist of Mastodon statuses. It can be seen from Table 3.4 that `misskey.io`, the instance which contributed the second most unique statuses to our corpus, is running Misskey and the third is running RSS Parrot. `Rss-parrot.net` and `sportsbots.xyz`, both not running Mastodon, are home to automated accounts only. Only four out of the top ten instances run Mastodon or compatible software (i.e., Fedibird). 10.6 million statuses originate from `misskey.io`, which is about half the amount of `mastodon.social`, despite not even crawling the former instance, contrary to the latter. Misskey seems to be popular in Japan, since we observed the main language on both of the two instances from Table 3.4 to be Japanese. Together with English, these are the only main languages on the top instances, as per our corpus.








Sensitive content makes up 3.8 % of unique statuses in our corpus. Some instances appear to not support or make use of this feature at all, as the percentage is equal to zero on three top instance. `Threads.net` apparently supports it, though 0.04 % of sensitive content seems particularly low next to other instances with a ratio greater zero, on which it ranges from 1.89 to 4.81 %. The ratio of statuses with attached media out of all statuses has a mean of 18.17 % across all instances, but differs greatly between the top instances, from zero up to 58.66 % on `sportsbots.xyz`. Spoilers are uncommon with a mean occurrence of 2.01 % in our corpus. While cards (i.e., URL previews) appear with very different frequencies on the top instances in our data, they








---

<sup>34</sup>Note that the set of candidate instances is about 5 % smaller than the set of instances usable for this analysis to ensure crawlability of sampled instances. The initial number of Mastodon instances with available NodeInfo data is 10,913.

<sup>35</sup>These instances do not necessarily contain all instances from our list fetched in December, 2023, which are still online.



**Table 3.4** Statistics of the top ten instances (according to first column) and of the set of all instances observed in our corpus. Columns are the number of unique statuses (= posts) originating from that instance, the software the instance is running (), main language () and the percentage of how often a property is present in all unique statuses from that instance(s). These properties are: `sensitive` (, makes `media_attachments` click-to-show), `media_attachments` (, e.g., images or videos), `spoiler_text` (, text displayed in front of a click-to-show spoiler, `content` is hidden in spoiler), `card` (URL preview), `reblog` () and `in_reply_to_id` (, reply).

Instance	Posts						Card		
mastodon.social	21.7M	Ma	en	2.79	25.13	1.13	39.95	15.47	13.76
misskey.io	10.6M	Mi	ja	4.81	11.54	1.66	8.87	1.31	1.89
rss-parrot.net	5.4M	RP	en	0.00	0.00	0.00	31.54	0.00	0.00
mstdn.jp	3.8M	Ma	ja	3.75	9.96	0.99	7.51	0.24	2.85
mstdn.social	3.3M	Ma	en	1.89	11.05	0.67	22.32	36.42	14.41
threads.net	2.7M	ud.	en	0.04	32.26	0.00	14.67	0.35	11.27
fedibird.com	2.6M	Fe	ja	2.78	9.08	1.35	14.80	1.52	6.46
sportsbots.xyz	2.6M	ud.	en	0.00	58.66	0.00	20.51	0.13	6.97
flipboard.com	2.5M	ud.	en	0.00	0.24	0.00	82.45	0.43	0.00
live-theater.net	2.4M	Mi	ja	2.13	6.15	2.04	4.90	0.00	0.46
24,026 instances	174.4M	–	en	3.80	18.17	2.01	24.71	13.73	11.88

<sup>1</sup> Ma: Mastodon; Mi: Misskey; RP: RSS Parrot; Fe: Fedibird (compatible Mastodon fork); ud.: undisclosed

are generally seen often, in almost every fourth status. The frequencies of reblogs and replies are nearly equivalent with 13.73 and 11.88 %, respectively, but with huge differences between instances, too. Accounts which are marked as a bot make up below 0.1 % of all accounts; the number of group accounts is negligibly small.

**Top Languages, Apps, Media Types, Hashtags** By far the two most used languages in our corpus are English with 34.71 % and Japanese with 23.39 %, measured by the number of statuses and their language field. Among the rest, German stands out a little, making up 4.46 %, other languages follow with 1.54 % or less. All numbers are displayed in Table 3.5. Among applications, the most important one is the Web client which is used to post 5.61 % of all statuses. The frequency of other applications does not go above 1.05 %. Note that only ca. 22 % of unique statuses declare an application. Images are

**Table 3.5** Top ten values of four fields in our corpus and the amount of their occurrence in all unique statuses. These fields are language (ISO 639 codes), application and—if applicable—operating system (🤖: Android, 🍏: iOS), hashtag, and type of media. Hashtag counts are absolute values, all other numbers percentages.

Language		Application		Media		Hashtag	
en	34.71	Web	5.61	image	80.78	news	31,363
ja	23.39	Mastodon (🤖)	1.05	unk.	13.12	press	31,037
de	4.46	dlvr.it	1.03	video	4.12	実況	26,370
fr	1.54	Jetpack	0.93	gifv	1.79	News	24,369
es	1.42	iembot	0.90	audio	0.18	nowplaying	22,308
zh	1.34	Mastodon (🍏)	0.84			nsfw	15,089
zh-CN	0.84	Tusky	0.72			37c3	11,868
pt	0.73	RSS投稿bot	0.60			bot	11,413
nl	0.71	MastoFeed	0.39			ukraine	10,211
ko	0.49	Ivory (🍏)	0.39			Ukraine	9,571

the most common form of media attachments, as they attribute for about four fifths of all media attachments. 13.12 % of media attachments are of unknown type though. The majority of top hashtags describe general categories like *news*, *press*, or *nowplaying*. Two hashtags stand out, because they describe events. These hashtags are *37c3* which refers to a relatively short event (the 37th Chaos Communication Congress) and *ukraine* which most likely has to do with the Russian invasion of the Ukraine.

# Chapter 4

## Conducting an IR Experiment using the Corpus

We conduct a baseline information retrieval experiment to preview the potential and limitations of our corpus. Besides the fixed set of documents, typical Cranfield paradigm experiments require topics, relevance judgments of documents and possibly different retrieval systems to compare. This chapter describes our approach and the results of the experiment.

### 4.1 Methodology

**Document Retrieval** Following the Cranfield paradigm, evaluating retrieval systems requires a set of information needs, also called topics. To derive the topics from real world information needs, we scanned various sources for search query keywords or hashtags that appeared frequently during the time we were building our corpus. These sources were Google Trends<sup>1</sup>, trending keywords and hashtags on the platform X<sup>2</sup>, trending Mastodon hashtags<sup>3</sup> and top weekly hashtags according to our own corpus. We condensed these keywords into 25 topics by formulating search queries from the keywords, adding descriptions and narratives. These topics are available in the git repository,<sup>4</sup> too.

To streamline the retrieval of relevant documents from the corpus, we have added the command `run-search-queries` to our Python package. The command parses the file containing all topics, creates one search query per topic per system, and sends the queries to the specified Elasticsearch installation. A

---

<sup>1</sup>e.g., via <https://trends.google.com/trends>

<sup>2</sup><https://archive.twitter-trending.com>

<sup>3</sup><https://mastodon.social/@TrendingHashtags>

<sup>4</sup><https://git.webis.de/code-teaching/theses/thesis-ernst/-/blob/main/data/topics.xml>

date range is added to each query to ensure that the set of documents is static across all queries. For each query, the program retrieves the ten documents with the highest relevance score according to this measure and stores them along with their ID in a JSONL file. We use Elasticsearch’s default relevance measure, which is Okapi BM25 [Rob+94].<sup>5</sup> The command creates two additional files, one in TrecRun format and the other in TrecQrel format (without relevance judgments), according to the file format imposed by TREC.<sup>6</sup>

We compare five different retrieval systems: (1) the local timeline of `mastodon.social` (the largest instance by status count in our corpus), (2) all statuses which were originally posted on `mastodon.social`, (3) the federated timeline of `mastodon.social`, (4) all statuses in the corpus from local timelines (`is_local == true`), and (5) all statuses in the corpus. Thus, 5 is a superset of every other system and 1 is a subset of every other system. 1, 3, and 4 inherently contain no duplicates, 2 and 5 are deduplicated upon retrieval to prevent the same original status with the highest relevance score from flooding a query, because it was crawled from different instances.

**Relevance Judgments** In a Cranfield-style experiment, documents of the corpus are usually judged before retrieval. We judged documents after retrieval instead, with the ratings 0 (not relevant) or 1 (relevant). Across the five retrieval systems, we judged all retrieved documents of the first five queries in regard to the corresponding topic. This is our first set of judgments we ranked the retrieval systems by. For all other queries, we judged exactly one (the uppermost) relevant document, which is the scenario considered in a paper by MacAvaney and Soldaini [MS23]. Utilizing their package `autoqrels`, we filled all holes in the relevance judgments. The manual judgments and those of `autoqrels` combined are the basis for another ranking of the retrieval systems. We furthermore deleted all but the uppermost relevant document judgment in the five fully manually judged queries per system, rejudged them with `autoqrels`, and ranked the systems again on the results. This last ranking allows to directly compare the rankings by our manual judgments and by the judgments of `autoqrels`. The process of (re-)judging document relevance and evaluating retrieval systems is also automated with our Python package and accessible via the command `infer-qrels`. We calculated the two recall-agnostic measures SDCG@10 [Jär+08] and P@10 and, because it is a TREC standard, nDCG@10 [JK02]. For calculating these measures, we resort to the *ir-measures* package by MacAvaney, Macdonald, and Ounis [MMO22].

---

<sup>5</sup><https://www.elastic.co/guide/en/elasticsearch/guide/current/relevance-intro.html>

<sup>6</sup><https://github.com/joaopalotti/trectools?tab=readme-ov-file#file-formats>

**Table 4.1** Evaluation of retrieval systems by the measures SDCG@10, P@10, and nDCG@10. Evaluated are three different sets of relevance judgments: all queries which were gaplessly judged by hand (M—Manual), the same queries from the *Manual* column but rejudged by autoqrels (R—Rejudged), and all queries from the *Manual* column plus all other queries judged by autoqrels (C—Combined). *ms* stands for *mastodon.social*.

System	SDCG@10			P@10			nDCG@10		
	M	R	C	M	R	C	M	R	C
<i>ms</i> local	0.57	0.56	0.51	0.52	0.50	0.44	0.85	0.90	0.86
<i>ms</i> origin	0.49	0.56	0.52	0.48	0.48	0.46	0.76	0.89	0.86
<i>ms</i> federated	0.47	0.52	0.55	0.48	0.48	0.51	0.73	0.83	0.86
All local statuses	0.45	0.39	0.51	0.42	0.32	0.46	0.76	0.79	0.86
All statuses	0.45	0.52	0.52	0.48	0.44	0.49	0.67	0.85	0.81

## 4.2 Results

As shown in Table 4.1, the ranking regarding the recall-agnostic measures in the manual and rejudged categories is generally the same, except in P@10 some scores are not distinct, whereas in SDCG@10 [Jär+08] they all are. The three systems around *mastodon.social* consistently score at least as high as the global systems. The “*mastodon.social* local” system, which could access the least amount of documents, always ranks first with manually judged or rejudged document relevance, with recall-agnostic scores ranging from 0.5 to 0.57. The other systems rank closely behind, except “All local statuses” which scores relatively low in the rejudged category with 0.39 in SDCG@10 and 0.32 in P@10.

In the combined category—where all queries are considered—“*mastodon.social* federated” ranks first in every measure. Regarding SDCG@10 and P@10, “All statuses” and “*mastodon.social* origin” follow in this order. “All local statuses” has the same score as “*mastodon.social* origin” in the P@10 measure, but is behind when considering SDCG@10. “*mastodon.social* local” ranks last regarding P@10 and shares its last place with “All local statuses” in the SDCG@10 measure. All scores of the measures SDCG@10 and P@10 are packed relatively close in the range of 0.42 to 0.57, with only two exceptions.

The system scores in the nDCG@10 measure are noticeably higher, ranging from 0.67 up to 0.9. In the combined category, “*mastodon.social* federated” also ranks first. Rank two to four are very close behind, and the system with access to all statuses scores lowest.

Since the ranking of the systems is the same for documents manually judged and rejudged by autoqrels regarding SDCG@10, Kendall's  $\tau$  coefficient for these two rankings is exactly 1. For the measure P@10, Kendall's  $\tau$  coefficient is a little lower, approximately 0.84.

# Chapter 5

## Discussion

**Corpus** We have described the architecture of our parallelized, polite crawler, and we invite everyone to contribute, inspect, or use it. The naming of the data fields that we store per status adheres to the Mastodon documentation. A few useful fields have been added to potentially simplify future experiments with the corpus. We sampled 1,000 Mastodon instances, including most of the large ones. Instances that appeared to have gone permanently offline during our crawl were replaced with instances closest in sampling weight. We have been crawling this set of instances for nearly 11 months and we plan to make it a full 12 months. The resulting corpus has been presented in several regards, particularly in terms of the instances which contributed the most statuses.

It has been shown that a considerable portion of statuses after deduplication originate from instances running software other than Mastodon. Examples are Misskey, a software appearing twice in the top ten instances, RSS Parrot, and the undisclosed software of `sportsbots.xyz` and Meta’s `threads.net`. By far the most common languages in the corpus are English (almost 35 %) and Japanese (ca. 23 %). Only around 4 % and 2 % of statuses are tagged with the third and fourth most used languages, respectively. Cards are very common, appearing in nearly every fourth status, which indicates that a lot of statuses contain a URL pointing to a web page. Media attachments—mostly images—are also quite frequent, as about 18 % of statuses contain at least one attachment. Despite occurring often in general, the quota of statuses containing cards or media attachments differs greatly between the top ten instances: from ca. 5 % to 82 % for cards and 0.2 % to 59 % for media attachments. Of all deduplicated statuses, about 14 % are reblogs and nearly 12 % are replies.

**Retrieval Experiment** We have conducted a baseline retrieval experiment using our corpus, with 25 original topics derived from popular keywords on Google Trends, the platform X, and Mastodon. Okapi BM25 [Rob+94] has

been used to estimate document relevance and, according to these relevance scores, the top ten results per query have been retrieved. After retrieval, we have judged document relevancy to the topics for five queries per system and for one relevant document per query for all other queries. All remaining holes have been filled utilizing autoqrels, a tool authored by MacAvaney and Soldaini [MS23]. We have also let autoqrels rejudge the document relevance of queries completely judged by us. Rank correlation of the five systems between manual and rejudged relevance ratings has been high, with a Kendall’s  $\tau$  coefficient of 1 for the SDCG@10 [Jär+08] and ca. 0.84 for the P@10 measure. In other words, the system ranking by autoqrels has shown good results, comparable to the results of the tool’s authors.

In this experiment, retrieval systems with a greater set of documents at hand do not generally rank higher. While the system with the smallest number of document available—`mastodon.social`’s local timeline—has the lowest score when considering all queries, the system that retrieved from all statuses only ranks only second to the system that had `mastodon.social`’s federated timeline at its disposal. We can only speculate about the reasons for this result. There might be a sweet spot for the amount of documents to search in when maximizing the relevancy of retrieved documents, or maybe a set of manually curated documents<sup>1</sup> leads to a higher average relevance than selecting another instance or timeline.

Another noteworthy observation is that the system which retrieved from all statuses posted on `mastodon.social` ranked above the system with only statuses of `mastodon.social`’s local timeline at hand. While the set should be identical for statuses created during the time of our crawl, the former system also retrieved from statuses created before our crawl.<sup>2</sup> Thus, the impact of changes in the timely dimension should be taken into account.

**Limitations and Future Work** Our corpus mainly contains statuses originating from Mastodon instances, but a considerable amount also stems from instances running other fediverse software. While this might even enable experiments regarding the entire fediverse, it adds to the complexity of Mastodon-only information retrieval or language processing, as a portion of Mastodon statuses require external data to unambiguously identify them as such. As shown in 3.2, Gathering Data about Instances, identifying the software running on an instance is not difficult if it integrates the NodeInfo standard, but

---

<sup>1</sup>The local timeline is a subset of the federated timeline. Additional statuses in the federated timeline are mostly from accounts external to an instance which were followed by a user on the original instance at the time of posting.

<sup>2</sup>Other instances fetched these older statuses and put them on their timeline during our crawl.



might be tedious for a larger number of software not integrating it. Filtering statuses by the Boolean `is_local` set to `true` will only yield Mastodon statuses, but also leave out a portion of all Mastodon statuses. Adding a field to our corpus indicating the instance software to every status is one possible task for future work.

Two other fields missing in our corpus are the activity metrics favorite count and reblog count. Since we usually crawl statuses right after their creation, these metrics are zero most of the time, or close to that. Furthermore, favorite and reblog counts are only accurate on the instance a status was originally posted on, because the home instance of a favoring or reblogging user only notifies the original instance, or additionally followers' instances in the case of a reblog (See Jambor [Jam23] for further explanation). These activity metrics could be added in a subsequent effort.

In our retrieval experiment, the margins that separate the systems are mostly relatively small. Hence, the findings should be taken with a grain of salt. The key point of this work is to enable research on information retrieval and natural language processing and to present the corpus. Only after several further experiments can there be findings that might be generalized. We have also observed two problems in our experiment: Retrieval has sometimes yielded duplicates or near-duplicates, even after deduplication. This might have been carbon copies or different bots posting the same news article. The second problem was retrieving statuses in languages other than English despite a corresponding filter, because users mislabeled the language of their statuses. Both problems likely had no impact on the overall outcome of the experiment as they affected all systems. These problems are not new (e.g., [BZ05; Efr11]), but we did not tackle them yet, and they should definitely be considered in follow-up experiments using the corpus.

One more point to consider is the centrality of the Mastodon network. There is a concentration of users and thus activity and content on the Mastodon developers' instance `mastodon.social`. Our corpus analysis shows that federation partially mitigates this effect, because the instances are far from isolated. Instead, a great amount of statuses is spread across instances, which suggests that users are generally interested in content on other instances. In terms of resource selection, the centrality entails a tendency to always select `mastodon.social` for high effectiveness, which might yield a higher number of relevant results, but not always the single most relevant documents. On the other hand, politeness dictates to put as little additional load as possible on very small instances in particular, as admins often privately pay for these. In conclusion, resource selection in this federated setting is a highly interesting, but also delicate topic that offers a lot of research potential.

# Chapter 6

## Conclusion

We have presented a first-of-its-kind comprehensive corpus of Mastodon statuses. It consists of 174 million deduplicated statuses and 3.6 billion statuses total, crawled from 1,081 instances over the course of 301 days. Aimed at the intersection of the previously separately researched areas of microblog and federated search, the corpus marks a starting point for future efforts in the field of federated microblog search. In the wake of the recent accelerated growth of the Mastodon userbase and Meta’s entry into the fediverse, in our view, this research area deserves more attention. We will provide privacy-preserving access to the corpus via TIREx, and the code of our polite and parallelized crawler is published alongside this work.

The analysis of the corpus has shown a great diversity of instances, particularly in terms of server software. All the statuses posted on these instances reached the timelines of the Mastodon instances that we crawled and are adding to the variety of our corpus, but also to the range of content lengths. Conducting the information retrieval experiment has given a first idea of the possibilities of the corpus and a few aspects to consider when designing a retrieval experiment that uses the corpus. In our findings, centrality is an important aspect to consider in resource selection.

Noteworthy limitations of the corpus are the difficult attribution of server software to statuses, first. Solely using the corpus, discriminating Mastodon statuses from other statuses is only possible for a portion of statuses. Attributing all statuses requires external data. Second, activity metrics are missing, because we crawled statuses right after they have been posted. We might add fields to tackle both of these limitations later. Nevertheless, the presented corpus offers an unprecedented opportunity to investigate federated microblog search.

# Bibliography

- [ÁC24] Lucía M. Álvarez-Crespo and Laura M. Castro. “A Galician Corpus for Misogyny Detection Online”. In: *Proceedings of the 16th International Conference on Computational Processing of Portuguese, PROPOR 2024, Santiago de Compostela, Galicia/Spain, 12-15 March, 2024*. Ed. by Pablo Gamallo, Daniela Claro, António J. S. Teixeira, Livy Real, Marcos García, Hugo Gonçalo Oliveira, and Raquel Amaro. Association for Computational Linguistics, 2024, pp. 22–31. URL: <https://aclanthology.org/2024.propor-1.3>.
- [Ana+23] Ishaku Hassan Anaobi, Aravindh Raman, Ignacio Castro, Haris Bin Zia, Damilola Ibosiola, and Gareth Tyson. “Will Admins Cope? Decentralized Moderation in the Fediverse”. In: *Proceedings of the ACM Web Conference 2023, WWW 2023, Austin, TX, USA, 30 April 2023 - 4 May 2023*. Ed. by Ying Ding, Jie Tang, Juan F. Sequeda, Lora Aroyo, Carlos Castillo, and Geert-Jan Houben. ACM, 2023, pp. 3109–3120. DOI: 10.1145/3543507.3583487.
- [Bon+24] Carlo Alberto Bono, Lucio La Cava, Luca Luceri, and Francesco Pierri. “An Exploration of Decentralized Moderation on Mastodon”. In: *Proceedings of the 16th ACM Web Science Conference, WEB-SCI 2024, Stuttgart, Germany, May 21-24, 2024*. Ed. by Luca Maria Aiello, Yelena Mejova, Oshani Seneviratne, Jun Sun, Sierra Kaiser, and Steffen Staab. ACM, 2024, pp. 53–58. DOI: 10.1145/3614419.3644016.
- [BV04] Chris Buckley and Ellen M. Voorhees. “Retrieval evaluation with incomplete information”. In: *SIGIR 2004: Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Sheffield, UK, July 25-29, 2004*. Ed. by Mark Sanderson, Kalervo Järvelin, James Allan, and Peter Bruza. ACM, 2004, pp. 25–32. DOI: 10.1145/1008992.1009000.

- [BZ05] Yaniv Bernstein and Justin Zobel. “Redundant documents and search effectiveness”. In: *Proceedings of the 2005 ACM CIKM International Conference on Information and Knowledge Management, Bremen, Germany, October 31 - November 5, 2005*. Ed. by Otthein Herzog, Hans-Jörg Schek, Norbert Fuhr, Abdur Chowdhury, and Wilfried Teiken. ACM, 2005, pp. 736–743. DOI: 10.1145/1099554.1099733.
- [Cer+18] Christophe Cerisara, Somayeh Jafaritazehjani, Adedayo Oluokun, and Hoa T. Le. “Multi-task dialog act and sentiment recognition on Mastodon”. In: *Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018*. Ed. by Emily M. Bender, Leon Derczynski, and Pierre Isabelle. Association for Computational Linguistics, 2018, pp. 745–754. URL: <https://aclanthology.org/C18-1063/>.
- [CGT21] Lucio La Cava, Sergio Greco, and Andrea Tagarelli. “Understanding the growth of the Fediverse through the lens of Mastodon”. In: *Appl. Netw. Sci.* 6.1 (2021), p. 64. DOI: 10.1007/S41109-021-00392-5.
- [CGT22] Lucio La Cava, Sergio Greco, and Andrea Tagarelli. “Information consumption and boundary spanning in Decentralized Online Social Networks: The case of Mastodon users”. In: *Online Soc. Networks Media* 30 (2022), p. 100220. DOI: 10.1016/J.OSNEM.2022.100220.
- [Dem+13] Thomas Demeester, Dolf Trieschnigg, Dong Nguyen, and Djoerd Hiemstra. “Overview of the TREC 2013 Federated Web Search Track”. In: *Proceedings of The Twenty-Second Text REtrieval Conference, TREC 2013, Gaithersburg, Maryland, USA, November 19-22, 2013*. Ed. by Ellen M. Voorhees. Vol. 500-302. NIST Special Publication. National Institute of Standards and Technology (NIST), 2013. URL: <http://trec.nist.gov/pubs/trec22/papers/FEDERATED.OVERVIEW.pdf>.
- [Dem+14] Thomas Demeester, Dolf Trieschnigg, Dong Nguyen, Djoerd Hiemstra, and Ke Zhou. “Overview of the TREC 2014 Federated Web Search Track”. In: *Proceedings of The Twenty-Third Text REtrieval Conference, TREC 2014, Gaithersburg, Maryland, USA, November 19-21, 2014*. Ed. by Ellen M. Voorhees and Angela Ellis. Vol. 500-308. NIST Special Publication. National Institute of Standards and Technology (NIST), 2014. URL: <http://trec.nist.gov/pubs/trec23/papers/overview-federated.pdf>.

- [Efr11] Miles Efron. “Information search and retrieval in microblogs”. In: *J. Assoc. Inf. Sci. Technol.* 62.6 (2011), pp. 996–1008. DOI: 10.1002/ASI.21512.
- [Frö+23a] Maik Fröbe, Jan Heinrich Reimer, Sean MacAvaney, Niklas Deckers, Simon Reich, Janek Bevendorff, Benno Stein, Matthias Hagen, and Martin Potthast. “The Information Retrieval Experiment Platform”. In: *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2023, Taipei, Taiwan, July 23-27, 2023*. Ed. by Hsin-Hsi Chen, Wei-Jou (Edward) Duh, Hen-Hsen Huang, Makoto P. Kato, Josiane Mothe, and Barbara Poblete. ACM, 2023, pp. 2826–2836. DOI: 10.1145/3539618.3591888.
- [Frö+23b] Maik Fröbe, Matti Wiegmann, Nikolay Kolyada, Bastian Grahm, Theresa Elstner, Frank Loebe, Matthias Hagen, Benno Stein, and Martin Potthast. “Continuous Integration for Reproducible Shared Tasks with TIRA.io”. In: *Advances in Information Retrieval - 45th European Conference on Information Retrieval, ECIR 2023, Dublin, Ireland, April 2-6, 2023, Proceedings, Part III*. Ed. by Jaap Kamps, Lorraine Goeuriot, Fabio Crestani, Maria Maistro, Hideo Joho, Brian Davis, Cathal Gurrin, Udo Kruschwitz, and Annalina Caputo. Vol. 13982. Lecture Notes in Computer Science. Springer, 2023, pp. 236–241. DOI: 10.1007/978-3-031-28241-6\_20.
- [He+23] Jiahui He, Haris Bin Zia, Ignacio Castro, Aravindh Raman, Nishanth Sastry, and Gareth Tyson. “Flocking to Mastodon: Tracking the Great Twitter Migration”. In: *Proceedings of the 2023 ACM on Internet Measurement Conference, IMC 2023, Montreal, QC, Canada, October 24-26, 2023*. Ed. by Marie-José Montpetit, Aris Leivadeas, Steve Uhlig, and Mobin Javed. ACM, 2023, pp. 111–123. DOI: 10.1145/3618257.3624819.
- [Jam23] Sebastian Jambor. *Understanding ActivityPub Part 3: The State of Mastodon*. 2023. URL: <https://seb.jambor.dev/posts/understanding-activitypub-part-3-the-state-of-mastodon/> (visited on 11/11/2024).
- [Jär+08] Kalervo Järvelin, Susan L. Price, Lois M. L. Delcambre, and Marianne Lykke Nielsen. “Discounted Cumulated Gain Based Evaluation of Multiple-Query IR Sessions”. In: *Advances in Information Retrieval, 30th European Conference on IR Research, ECIR 2008, Glasgow, UK, March 30-April 3, 2008. Proceedings*. Ed. by Craig

- Macdonald, Iadh Ounis, Vassilis Plachouras, Ian Ruthven, and Ryan W. White. Vol. 4956. Lecture Notes in Computer Science. Springer, 2008, pp. 4–15. DOI: 10.1007/978-3-540-78646-7\_4.
- [Jeo+24] Ujun Jeong, Paras Sheth, Anique Tahir, Faisal Alatawi, H. Russell Bernard, and Huan Liu. “Exploring Platform Migration Patterns between Twitter and Mastodon: A User Behavior Study”. In: *Proceedings of the Eighteenth International AAAI Conference on Web and Social Media, ICWSM 2024, Buffalo, New York, USA, June 3-6, 2024*. Ed. by Yu-Ru Lin, Yelena Mejova, and Meeyoung Cha. AAAI Press, 2024, pp. 738–750. DOI: 10.1609/ICWSM.V18I1.31348.
- [JK02] Kalervo Järvelin and Jaana Kekäläinen. “Cumulated gain-based evaluation of IR techniques”. In: *ACM Trans. Inf. Syst.* 20.4 (2002), pp. 422–446. DOI: 10.1145/582415.582418.
- [KS18] Ram Kumar and S. C. Sharma. “Information Retrieval System: An Overview, Issues, and Challenges”. In: *Int. J. Technol. Diffusion* 9.1 (2018), pp. 1–10. DOI: 10.4018/IJTD.2018010101.
- [LE13] Jimmy Lin and Miles Efron. “Overview of the TREC-2013 Microblog Track”. In: *Proceedings of The Twenty-Second Text REtrieval Conference, TREC 2013, Gaithersburg, Maryland, USA, November 19-22, 2013*. Ed. by Ellen M. Voorhees. Vol. 500-302. NIST Special Publication. National Institute of Standards and Technology (NIST), 2013. URL: <http://trec.nist.gov/pubs/trec22/papers/MB.OVERVIEW.pdf>.
- [Lem+18] Christine Lemmer-Webber, Jessica Tallon, Erin Shepherd, Amy Guy, and Evan Prodromou. *ActivityPub*. Ed. by Christine Lemmer-Webber and Jessica Tallon. World Wide Web Consortium, 2018. URL: <https://www.w3.org/TR/2018/REC-activitypub-20180123/>.
- [Lin+14] Jimmy Lin, Yulu Wang, Miles Efron, and Garrick Sherman. “Overview of the TREC-2014 Microblog Track”. In: *Proceedings of The Twenty-Third Text REtrieval Conference, TREC 2014, Gaithersburg, Maryland, USA, November 19-21, 2014*. Ed. by Ellen M. Voorhees and Angela Ellis. Vol. 500-308. NIST Special Publication. National Institute of Standards and Technology (NIST), 2014. URL: <https://trec.nist.gov/pubs/trec23/papers/overview-microblog.pdf>.

- [Lin+15] Jimmy Lin, Miles Efron, Garrick Sherman, Yulu Wang, and Ellen M. Voorhees. “Overview of the TREC-2015 Microblog Track”. In: *Proceedings of The Twenty-Fourth Text REtrieval Conference, TREC 2015, Gaithersburg, Maryland, USA, November 17-20, 2015*. Ed. by Ellen M. Voorhees and Angela Ellis. Vol. 500-319. NIST Special Publication. National Institute of Standards and Technology (NIST), 2015. URL: <https://trec.nist.gov/pubs/trec24/papers/Overview-MB.pdf>.
- [McC+12] Richard McCreadie, Ian Soboroff, Jimmy Lin, Craig Macdonald, Iadh Ounis, and Dean McCullough. “On building a reusable Twitter corpus”. In: *The 35th International ACM SIGIR conference on research and development in Information Retrieval, SIGIR ’12, Portland, OR, USA, August 12-16, 2012*. Ed. by William R. Hersh, Jamie Callan, Yoelle Maarek, and Mark Sanderson. ACM, 2012, pp. 1113–1114. DOI: 10.1145/2348283.2348495.
- [MMJ13] Andrew James McMinn, Yashar Moshfeghi, and Joemon M. Jose. “Building a large-scale corpus for evaluating event detection on twitter”. In: *22nd ACM International Conference on Information and Knowledge Management, CIKM’13, San Francisco, CA, USA, October 27 - November 1, 2013*. Ed. by Qi He, Arun Iyengar, Wolfgang Nejdl, Jian Pei, and Rajeev Rastogi. ACM, 2013, pp. 409–418. DOI: 10.1145/2505515.2505695.
- [MMO22] Sean MacAvaney, Craig Macdonald, and Iadh Ounis. “Streamlining Evaluation with ir-measures”. In: *Advances in Information Retrieval - 44th European Conference on IR Research, ECIR 2022, Stavanger, Norway, April 10-14, 2022, Proceedings, Part II*. Ed. by Matthias Hagen, Suzan Verberne, Craig Macdonald, Christin Seifert, Krisztian Balog, Kjetil Nørkvåg, and Vinay Setty. Vol. 13186. Lecture Notes in Computer Science. Springer, 2022, pp. 305–310. DOI: 10.1007/978-3-030-99739-7\_38.
- [MS23] Sean MacAvaney and Luca Soldaini. “One-Shot Labeling for Automatic Relevance Estimation”. In: *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2023, Taipei, Taiwan, July 23-27, 2023*. Ed. by Hsin-Hsi Chen, Wei-Jou (Edward) Duh, Hen-Hsen Huang, Makoto P. Kato, Josiane Mothe, and Barbara Poblete. ACM, 2023, pp. 2230–2235. DOI: 10.1145/3539618.3592032.
- [Nav+11] Nasir Naveed, Thomas Gottron, Jérôme Kunegis, and Arifah Che Alhadi. “Searching microblogs: coping with sparsity and document

- quality”. In: *Proceedings of the 20th ACM Conference on Information and Knowledge Management, CIKM 2011, Glasgow, United Kingdom, October 24-28, 2011*. Ed. by Craig Macdonald, Iadh Ounis, and Ian Ruthven. ACM, 2011, pp. 183–188. DOI: 10.1145/2063576.2063607.
- [Oun+11] Iadh Ounis, Craig Macdonald, Jimmy Lin, and Ian Soboroff. “Overview of the TREC 2011 Microblog Track”. In: *Proceedings of The Twentieth Text REtrieval Conference, TREC 2011, Gaithersburg, Maryland, USA, November 15-18, 2011*. Ed. by Ellen M. Voorhees and Lori P. Buckland. Vol. 500-296. NIST Special Publication. National Institute of Standards and Technology (NIST), 2011. URL: <https://trec.nist.gov/pubs/trec20/papers/MICROBLOG.OVERVIEW.pdf>.
- [Ram+19] Aravindh Raman, Sagar Joglekar, Emiliano De Cristofaro, Nishanth Sastry, and Gareth Tyson. “Challenges in the Decentralised Web: The Mastodon Case”. In: *Proceedings of the Internet Measurement Conference, IMC 2019, Amsterdam, The Netherlands, October 21-23, 2019*. ACM, 2019, pp. 217–229. DOI: 10.1145/3355369.3355572.
- [Rob+94] Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. “Okapi at TREC-3”. In: *Proceedings of The Third Text REtrieval Conference, TREC 1994, Gaithersburg, Maryland, USA, November 2-4, 1994*. Ed. by Donna K. Harman. Vol. 500-225. NIST Special Publication. National Institute of Standards and Technology (NIST), 1994, pp. 109–126. URL: <http://trec.nist.gov/pubs/trec3/papers/city.ps.gz>.
- [Roi+20] Kevin Roitero, Andrea Brunello, Giuseppe Serra, and Stefano Mizzaro. “Effectiveness evaluation without human relevance judgments: A systematic analysis of existing methods and of their combinations”. In: *Inf. Process. Manag.* 57.2 (2020), p. 102149. DOI: 10.1016/J.IPM.2019.102149.
- [Sob+12] Ian Soboroff, Iadh Ounis, Craig Macdonald, and Jimmy Lin. “Overview of the TREC-2012 Microblog Track”. In: *Proceedings of The Twenty-First Text REtrieval Conference, TREC 2012, Gaithersburg, Maryland, USA, November 6-9, 2012*. Ed. by Ellen M. Voorhees and Lori P. Buckland. Vol. 500-298. NIST Special Publication. National Institute of Standards and Technology (NIST), 2012. URL: <http://trec.nist.gov/pubs/trec21/papers/MICROBLOG12OVERVIEW.pdf>.



- [SS11] Milad Shokouhi and Luo Si. “Federated Search”. In: *Found. Trends Inf. Retr.* 5.1 (2011), pp. 1–102. DOI: 10.1561/15000000010.
- [TCH18] Jan Trienes, Andrés Torres Cano, and Djoerd Hiemstra. *Recommending Users: Whom to Follow on Federated Social Networks*. arXiv 1811.09292. 2018. DOI: 10.48550/arXiv.1811.09292.
- [UKL24] Shivani Upadhyay, Ehsan Kamaloo, and Jimmy Lin. *LLMs Can Patch Up Missing Relevance Judgments in Evaluation*. arXiv 2405.04727. 2024. DOI: 10.48550/arXiv.2405.04727.
- [Voo01] Ellen M. Voorhees. “The Philosophy of Information Retrieval Evaluation”. In: *Evaluation of Cross-Language Information Retrieval Systems, Second Workshop of the Cross-Language Evaluation Forum, CLEF 2001, Darmstadt, Germany, September 3-4, 2001, Revised Papers*. Ed. by Carol Peters, Martin Braschler, Julio Gonzalo, and Michael Kluck. Vol. 2406. Lecture Notes in Computer Science. Springer, 2001, pp. 355–370. DOI: 10.1007/3-540-45691-0\_34.
- [Wie+24] Matti Wiegmann, Jan Heinrich Reimer, Maximilian Ernst, Martin Potthast, Matthias Hagen, and Benno Stein. “A Mastodon Corpus to Evaluate Federated Microblog Search”. In: *Proceedings of the first International Workshop on Open Web Search co-located with the 46th European Conference on Information Retrieval ECIR 2024, Glasgow, UK, March 28, 2024*. Ed. by Sheikh Mastura Farzana, Maik Fröbe, Gijs Hendriksen, Michael Granitzer, Djoerd Hiemstra, Martin Potthast, and Saber Zerhoubi. Vol. 3689. CEUR Workshop Proceedings. CEUR-WS.org, 2024, pp. 37–49. URL: [https://ceur-ws.org/Vol-3689/WOWS\\_2024\\_paper\\_5.pdf](https://ceur-ws.org/Vol-3689/WOWS_2024_paper_5.pdf).
- [WKR24] Xinyu Wang, Sai Dileep Koneru, and Sarah Rajtmajer. *The Failed Migration of Academic Twitter*. arXiv 2406.04005. 2024. DOI: 10.48550/arXiv.2406.04005.
- [YA06] Emine Yilmaz and Javed A. Aslam. “Estimating average precision with incomplete and imperfect judgments”. In: *Proceedings of the 2006 ACM CIKM International Conference on Information and Knowledge Management, Arlington, Virginia, USA, November 6-11, 2006*. Ed. by Philip S. Yu, Vassilis J. Tsotras, Edward A. Fox, and Bing Liu. ACM, 2006, pp. 102–111. DOI: 10.1145/1183614.1183633.

- [YM20] Jingyuan Yu and Juan Muñoz-Justicia. “A Bibliometric Overview of Twitter-Related Studies Indexed in Web of Science”. In: *Future Internet* 12.5 (2020), p. 91. DOI: 10.3390/FI12050091.
- [ZGR18] Matteo Zignani, Sabrina Gaito, and Gian Paolo Rossi. “Follow the "Mastodon": Structure and Evolution of a Decentralized Online Social Network”. In: *Proceedings of the Twelfth International Conference on Web and Social Media, ICWSM 2018, Stanford, California, USA, June 25-28, 2018*. AAAI Press, 2018, pp. 541–551. DOI: 10.1609/icwsm.v12i1.14988.
- [Zig+22] Matteo Zignani, Christian Quadri, Alessia Galdeman, Sabrina Gaito, and Gian Paolo Rossi. “Statement of Removal. Mastodon Content Warnings: Inappropriate Contents in a Microblogging Platform”. In: *Proceedings of the Thirteenth International Conference on Web and Social Media, ICWSM 2019, Munich, Germany, June 11-14, 2019*. Ed. by Jürgen Pfeffer, Ceren Budak, Yu-Ru Lin, and Fred Morstatter. AAAI Press, 2022, pp. 639–645. URL: <https://ojs.aaai.org/index.php/ICWSM/article/view/22003>.
- [ZLG20] Diana Zulli, Miao Liu, and Robert W. Gehl. “Rethinking the "social" in "social media": Insights into topology, abstraction, and scale on the Mastodon social network”. In: *New Media Soc.* 22.7 (2020). DOI: 10.1177/1461444820912533.



## Declaration of Academic Integrity

1. I hereby confirm that this work – or in case of group work, the contribution for which I am responsible and which I have clearly identified as such – is my own work and that I have not used any sources or resources other than those referenced.

I take responsibility for the quality of this text and its content and have ensured that all information and arguments provided are substantiated with or supported by appropriate academic sources. I have clearly identified and fully referenced any material such as text passages, thoughts, concepts or graphics that I have directly or indirectly copied from the work of others or my own previous work. Except where stated otherwise by reference or acknowledgement, the work presented is my own in terms of copyright.

2. I understand that this declaration also applies to generative AI tools which cannot be cited (hereinafter referred to as 'generative AI').

I understand that the use of generative AI is not permitted unless the examiner has explicitly authorized its use (Declaration of Permitted Resources). Where the use of generative AI was permitted, I confirm that I have only used it as a resource and that this work is largely my own original work. I take full responsibility for any AI-generated content I included in my work.

Where the use of generative AI was permitted to compose this work, I have acknowledged its use in a separate appendix. This appendix includes information about which AI tool was used or a detailed description of how it was used in accordance with the requirements specified in the examiner's Declaration of Permitted Resources.

I have read and understood the requirements contained therein and any use of generative AI in this work has been acknowledged accordingly (e.g. type, purpose and scope as well as specific instructions on how to acknowledge its use).

3. I also confirm that this work has not been previously submitted in an identical or similar form to any other examination authority in Germany or abroad, and that it has not been previously published in German or any other language.
4. I am aware that any failure to observe the aforementioned points may lead to the imposition of penalties in accordance with the relevant examination regulations. In particular, this may include that my work will be classified as deception and marked as failed. Repeated or severe attempts to deceive may also lead to a temporary or permanent exclusion from further assessments in my degree programme.

.....  
Place and date

.....  
Signature