Leveraging Wikipedia Page Edits for Analytical Processing

Tim Träris
Faculty of Computer Science
Furtwangen University
Furtwangen, Germany
tim.jannes.traeris@hs-furtwangen.de

Maxim Balsacq
Faculty of Computer Science
Furtwangen University
Furtwangen, Germany
maxim.balsacq@hs-furtwangen.de

***Abstract*—Wikipedia is the largest free encyclopedia and one of the most popular websites worldwide. Analyzing user activity within this encyclopedic ecosystem represents unique opportunities for academic research and analysis. For this reason, this work is fundamentally concerned with obtaining and processing real-time article edit streams from Wikipedia. In this regard, we leverage the Wikimedia EventStreams API and propose a general-purpose event pipeline allowing for further processing of observed page edits. In the suggested pipeline, events are ingested and transported via an Apache Kafka cluster and inserted into a ClickHouse database for storage and analysis. Finally, we confirm the viability of our design by exploring several exemplary analytical use cases.**

***Index Terms*—Event Stream Processing, Analysis, Wikipedia, Apache Kafka, ClickHouse**

I. Introduction

Wikipedia is the world's largest free encyclopedia and as of January 2021 one of the 15 most popular websites worldwide ranked by Alexa [1]. It is funded primarily through an annual call for donations and hosted by the non-profit Wikimedia Foundation. As an open and collaborative encyclopedia, Wikipedia constitutes a multilingual platform of knowledge and information sharing maintained by a community of millions of voluntary authors. Anybody can easily contribute to Wikipedia by adding, editing and reviewing its contents.

Due to the vast amount of information openly available, Wikipedia is often popularly ranked in the results of search engines with an estimated 74% of its total traffic originating from referrals of search results [1]. Naturally, Wikipedia also embodies an unique vantage point for analyses of user behaviour and interests. Fortunately, detailed information regarding user activities on Wikipedia i.e. page edits is openly published as an event-based stream by services provided by the Wikimedia Foundation. In this paper, we explore how this data stream can be captured, processed and analyzed in order to enable research and contribute to the understandig of the Wikipedia ecosystem.

In the following, we explain the basic concepts of event-based stream processing and introduce our data source.

A. Event Stream Processing

Event stream processing (ESP) is a general term describing methods, technologies and tools used to continuously process events as they arrive as an ongoing series in real-time [2]. Each event constitutes some kind of data point which is extracted and processed by the system in order to deduce and gain information. With the increasing amount of big data applications, ESP has been a key solution when batch processing is no longer feasible. In this regard, ESP is an enabling technology for research purposes aiming to analyze and understand real-time events.

However, use cases for ESP extend far beyond research and academia. In domains like business intelligence and sales, the goal is usually to take action based on the obtained knowledge i.e. to identify meaningful incidents and anomalies and respond to them as quickly as possible. Further, gathered information also enables predictions e.g. for upcoming maintenance [3, 4] or stock market activity [5, 6].

B. Data Source: Wikimedia EventStreams

The Wikimedia Foundation mainly provides two different services exposing events for their wikis: an Internet Relay Chat (IRC) based service running at irc.wikimedia.org [7] and a HTTP-based service called EventStreams [8] running at https://stream.wikimedia.org. Given that EventStreams is the successor and is set to replace irc.wikimedia.org in the future, we will focus on and use the EventStreams service going forward.

EventStreams is a web-based service exposing events as a continuous HTTP stream which is pushed to its clients. This push technology is implemented by the Server-Sent Events (SSE) [9] protocol using chunked transfer encoding - a mechanism which divides the data stream into independent chunks. These chunks are sent out to all connected HTTP clients. As illustrated in figure 1, the EventStreams service is designed similar to a publish-subscribe pattern: Clients can connect (subscribe) to the various offered streams and get served continuously as events occur and get published by Wikimedia services.

The EventStreams service offers a variety of different streams for different kinds of events, i.e. page creation, page deletion, page edits, page properties change, page restore, page move among others. For this paper, we will solely focus on page edits as this is by far the most frequent user activity seen on Wikipedia. We will also limit our investigation on page

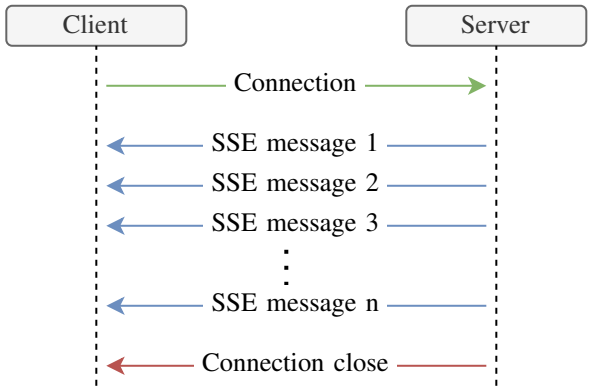

Fig. 1. Using the Server-Sent Events (SSE) protocol, clients connect to the server and continuously receive a stream of messages until the connection is closed.

edits of the English and German versions of Wikipedia available at en.wikipedia.org and de.wikipedia.org respectively. However, all shown concepts and used methods are universally applicable and can be used with all wikis of the Wikimedia Foundation.

Events are served to clients in a JavaScript Object Notation (JSON) format. Therefore, event data is human-readable and easily serializable by default. While it is possible for clients to consume events with basic CLI tools like `curl`, SSE client libraries usually enable a more sophisticated approach by providing functionality for filtering and further processing of received data.

II. Related Work

In "MJ no more: Using Concurrent Wikipedia Edit Spikes with Social Network Plausibility Checks for Breaking News Detection" (2013) [10], Steiner et al. develop an application that detects page edit spikes on Wikipedia and correlates them with social network activities. In this regard, the application consumes page edit events of different languages using the IRC service and groups them by topic. The authors argue that an event could be classified as potential breaking news whenever the respective topic group reaches five edits by at least two concurrent editors with less than 60 seconds between edits and less than 240 seconds since its last edit. Finally, identified breaking news candidates are cross-checked with a full-text search on Twitter, Facebook and Google+ in order to validate them. Steiner et al. confirm the assumption that breaking news are regularly discussed on social media and conclude that their developed prototype delivers promising results. For future work, the authors suspect that article categories could be strong indicators of news events as well - a conjecture that motivated us to further investigate user behaviour and these categories in section IV-B.

In "Real-time Event-based News Suggestion for Wikipedia Pages from News Streams" (2018) [11], Lyu et al. remark that a major share of Wikipedia articles is incomplete and/or lacking external references. In order to improve the verifiability of content on Wikipedia, the authors propose an algorithm that relates external news articles to Wikipedia articles published as current events. As a result, Lyu et al. are able to suggest references from authoritative news media and therefore enrich corresponding articles on Wikipedia.

Both of these papers demonstrate the relevance of Wikipedia metadata for analyses and research purposes. In the following sections, we propose a processing pipeline for EventStreams messages and subsequently showcase how our approach can benefit future research and analytical use cases.

III. Event Processing Pipeline Implementation

During this section, we will develop a prototypical event pipeline in order to leverage the EventSteams API for analytical processing. Our implementation will be based on Python scripts, Apache Kafka and ClickHouse. Figure 2 provides an overview of the pipeline workflow. All steps in this processing chain allow for near real-time evaluation and event analysis.

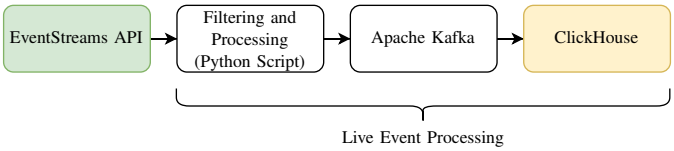

Fig. 2. EventStreams API messages are filtered, processed and ingested into Apache Kafka for consumption by ClickHouse.

A. Data Retrieval and Information Extraction

As introduced in section I-B, the Wikipedia EventStreams API can be tapped into by any basic HTTP client with support for SSE. Since the SSE protocol as well as JSON messages are common standards in the field, a plethora of tools and libraries for most programming languages are available to consume event messages. For our implementation, we use Pywikibot [12], a Python library developed by the Wikimedia Foundation. At its core, this library is a collection of tools for developing bots that access Wikimedia wikis and their APIs. Further, the framework also powers various automated tasks on Wikipedia as it allows to read and edit content. We specifically chose this library as it offers built-in filter capabilities for the EventStreams service and is directly maintained by its creators.

As mentioned previously, we focus on the retrieval of page edit events for the scope of this paper. In particular, we filter, capture and extract the following fields for each edit on the German and English versions of Wikipedia:

- `timestamp`: Naturally, each event contains an ISO 8601 timestamp. All timestamps are transmitted with second precision.
- `user`: The user responsible for the page edit. In case the page has been edited without creating an account, the users IPv4 or IPv6 address is recorded instead.
- `bot`: Whether the user is classified as a bot account.
- `minor`: Whether the edit is considered to be minor edit. According to Wikipedia guidelines [13], minor edits are defined as alterations that do not change the factual content of the article and therefore only include superficial

changes. Minor edits are manually denoted by the user via checkbox.
- `title`: The title of the article.
- `comment`: Edits optionally contain a comment describing the changes. Comments can be worded by the user or are automatically generated in some cases e.g. rollback of revisions or bot actions.
- `meta["uri"]`: The full URI of the article which is used to access it via web browser.
- `length`: The length in bytes. In order to properly describe the changes, a value for old and new length is transmitted.
- `revision`: Each version of an article is assigned an unique revision ID. As for the length, we capture both the old and new revision ID.

All SSE messages arriving at the client are UTF-8 byte-encoded and sent to the local Apache Kafka instance.

B. Apache Kafka

Apache Kafka [14] is an open source distributed streaming platform originally developed by LinkedIn and now maintained by the non-profit Apache Software Foundation. It is based on a distributed transaction log optimized for failure tolerance and scalability while enabling processing and storage of data streams. In this regard, Kafka provides various interfaces and libraries to write and read data to and from Kafka clusters respectively. Clients writing to Kafka are called producers, while clients reading from Kafka are referred to as consumers. On Kafka brokers, data streams are processed and stored as topics containing messages as key-value pairs labeled with timestamps.

In addition, Kafka supports exactly-once stream processing and offers built-in protection against duplication in terms of writing and processing. This idempotent writing process is implemented by unique producer identifiers and by attaching sequence numbers to each message. Further, the atomicity of read-process-write transactions can be ensured at all times due to Kafka's distributed transaction log. The messages of completed and aborted transactions are internally labeled with commit and abort markers respectively. Subsequently, only successfully transmitted and processed messages are presented to consumers. As a result, this allows us to easily implement delivery guarantees for our event pipeline while also enabling future scale out by adding Kafka brokers.

Regarding the implementation of our proof-of-concept, we use Kafka as a message queue system between data collection (Python script) and data analysis (ClickHouse). This allows us to buffer the incoming data stream and provides opportunities for real-time processing as well as a delayed replay of the stream. While a single Kafka broker suffices for our use case, relying on Kafka for this step in the processing chain allows easy scale out in the future by expanding the cluster. Further, Kafka's processing capabilities could be leveraged e.g. to redirect certain messages to different topics resulting in sophisticated event distribution for complex use cases.

C. ClickHouse

ClickHouse is an open-source column-oriented database management system (DBMS) developed by Yandex. From its core, it has been specifically designed to deliver superior performance, low latency and nearly instant results to meet the demands of online analytical processing (OLAP) queries [15]. Queries can be easily shaped using a declarative SQL-based language. Data is stored on disk, indexed and sorted by primary key. Compression algorithms can be assigned server-wide or per column to increase storage efficiency. In this regard, data is not only stored by columns but as parts of columns (vectors) and processed as such. This vector engine plays a key role in parallelization, as large queries are parallelized automatically across multiple cores and even multiple servers in a cluster setup.

Besides its general focus on analytic workloads, ClickHouse also offers functionality for real-time processing of newly inserted data sets. This is implemented by materialized views that watch a specific table and store data transformed by a corresponding query. As a result, whenever data sets are inserted into this table, the materialized view is updated accordingly. Querying the view therefore always returns an up-to-date result for the materialized view query. Further, ClickHouse supports external dictionaries that reside on disk or in memory and provide key-value lookups on query runtime. Consequently, dictionaries enable enrichment of EventStreams messages with additional data from external sources as showcased later in section IV-C.

For our proof of concept, we utilize ClickHouse as an analytical tool and long-term storage for observed EventStreams API messages. This allows us to perform ad-hoc queries examining stored as well as real-time data. However, ClickHouse could be replaced with any suitable database, analytical platform or custom scripts that are able to consume data sets from Apache Kafka.

IV. Exemplary Use Cases

In this section, we test the viability of our pipeline design by exploring potential analytical use cases. For our analyses, we examine Wikipedia page edit events between the 1st October and the 31st December 2020. During this period, we recorded 15.359.372 and 2.560.742 edit events for the English and German editions of Wikipedia respectively.

A. General Statistics

Using the collected metadata events, various types of statistics can be generated which help to better understand the behavior of users and bots. This in turn may help to design tools which try to detect harmful activity on Wikipedia.

1) User Participation: Counting edits for each user results in statistics which can help to visualize how the work of certain users has influenced Wikipedia. Figure 3 displays the percentage of edits contributed by the most active users. For example, we found that in the English Wikipedia, 0.001% of human authors have contributed about 5% of all edits. This amounts to the same contribution percentage as the bottom

60% of users, as they mostly contribute only one edit each. When including bots in this figure, the distribution is even more extreme: the top 0.001% of users contribute 15% of all edits. The German Wikipedia on the other hand seems to rely a lot less on bots: When including bots in the calculation, the percentage of edits performed by the top 0.001% decreases, indicating that there are more active human users than bots among the top 0.001% of users.

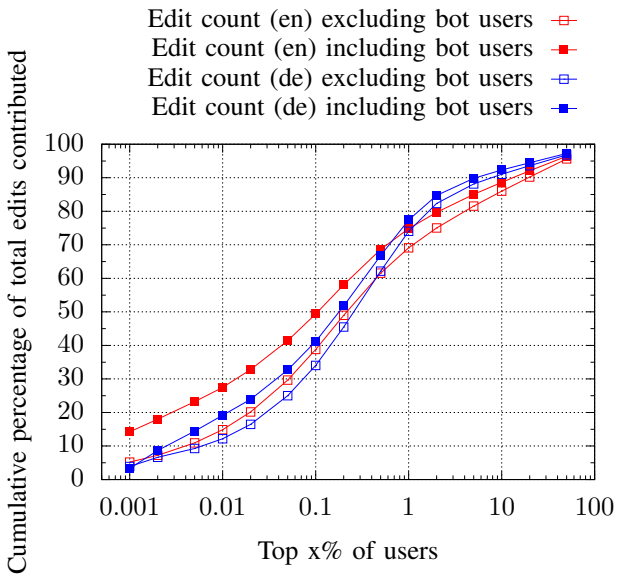

Fig. 3. Wikipedia contribution dispersion by users

However, it should be noted that pure edit counts may not accurately portray how much a user has contributed. Users contributing larger edits will take longer to create each edit, while users using automated tools, e.g. to review and revert vandalism, commonly have higher edit counts with a lower contribution per edit.

2) Behaviour of Bots: Wikipedia uses bots to improve articles e.g. to combat harmful user activity and to keep links and references up to date. Any user may pose a request to the Wikipedia Bot Approval Group (BAG) in order to register a user account as a bot [16]. The BAG checks the purpose and quality of the bot before approving or denying the request. A user account approved as a bot may perform edits which are hidden in the recent changes [17] and show up with the bot flag set in the EventStreams feed. Obtaining the bot flag also allows bots to bypass rate-limiting rules, thus execute edits more frequently.

For our analyzed time period, we observed that about 16.52% on the English and 13.27% of edits on the German Wikipedia are performed by users labeled with a bot flag. Some bots also perform edits with the bot flag unset to enable manual review of the automated edits. However, not all automated edits are performed by bots. Numerous tools exist to speed up the process of editing, categorizing, and reviewing articles as well as to quickly find and eliminate vandalism.

Although edits by bots may be useful, it is always up to the authors to declare their purpose and control their usage. This means that it is also possible for malicious users to write their own bots, e.g. to manipulate and automatically revert certain edits. While it may not be easy to determine the purpose of a bot, it is possible to detect bots unless the authors have taken care to obfuscate their use:

- Period of activity: Assuming that an account belongs to a human, the account cannot be active 24/7. By checking which users are editing the whole day without breaks, we can easily identify bots. However, this detection method also returns human users who contribute a lot.
- Editing speed: While humans need some time to create changes to articles, bots face no such limitations. It is however possible for humans to use tools to create many changes in a short time. These semi-automated edits are difficult to distinguish from bot activity.

3) Vandalism statistics: Given the vast amount of edit events, detecting vandalism reliably becomes a pressing issue for Wikipedia administration. Harmful edits must be reverted in a timely manner to prevent the spread of misinformation. Wikipedia allows to revert an edit by applying the diff of the vandalizing edit in reverse. This is called a revert. The revert itself is a regular edit with a user-supplied comment. This comment is automatically prefixed e.g. with ”Undid revision” followed by the revision id. By searching the comments of edits for ”Undid revision” and ”revert”, and assuming each edit reverted a vandalizing edit, it is possible to generate statistics indicating how many vandalizing edits exist. However, the exact search terms to find these edits vary between the different Wikipedia editions due to the edits comments being worded in their respective language.

Figure 4 illustrates the percentage of reverts executed each day for both English and German Wikipedia. Although we can observe a similar pattern for the two wikis, the English edition of Wikipedia generally sees an increased revert rate. One interesting detail is the lows in both graphs, which mostly occur on the weekends. Upon investigation, we found that the overall edit count is higher when these dips occur, suggesting that the additional edits generally are of a higher quality.

Further, filtering edits by titles in a specific category allows to detect if edits in specific categories are more or less susceptible to vandalism. This information could be used to create stricter page protection constraints on categories. However, such interpretation must be considered carefully, since a category may only include very few edits. Consequently, a few additional vandalizing edits may have a strong impact on the percentage of reverts in the category.

B. Analyzing User Interests

Analyzing the interests of a user can be of great use, e.g. to recommend a Wikipedia article or category they could join and therefore edit pages in their specific area of knowledge and interest. It could also be used to detect when a user is editing something outside his usual area of interest.

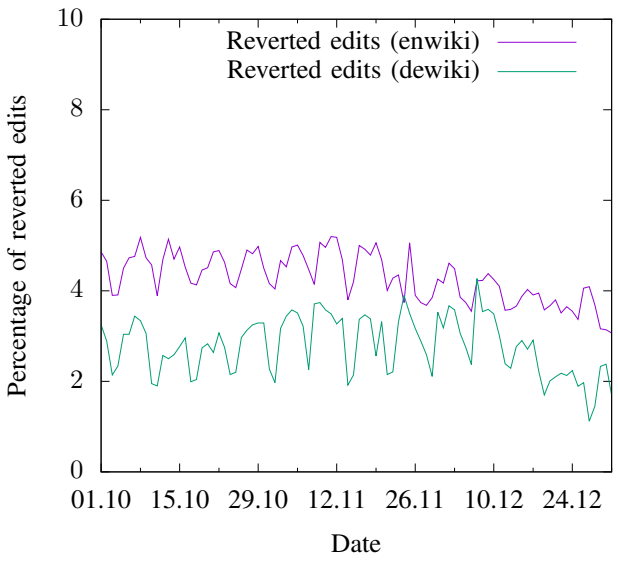

Fig. 4. Count of reverted edits over time

An analysis of user interests may be performed by checking the categories of a page. Naturally, if a user performs multiple edits in a specific category, this category may be of interest to the user. Unfortunately, the categories of an article are set by editing the article itself, meaning that the categories of an article can not solely be determined by using the EventStreams data. Therefore, it is not easily possible to analyze the categories of new articles.

However, SQL dumps of all categories existing on a Wikipedia instance are published on a weekly basis. This dump includes the `categorylinks` table [18], which maps a page id to the list of categories it belongs to. Together with the `page_table` table, a mapping of each page or category to a list of sub- and super-categories can be created. Using this information, it is possible to classify new edits on existing pages or retroactively apply the category information to old EventStreams data.

Unfortunately, importing this dump in order to use it in combination with our proposed event processing pipeline was not without problems. The `categorylinks` table for the English Wikipedia is only obtainable in the form of a 23GB MySQL dump, which - in our case - was projected to take more than 18 hours to import. The performance of MySQL further degraded as data was imported, making an import impractical. Therefore, we wrote a script to extract only the necessary columns and generate a CSV file, which can be easily imported into other database systems (such as ClickHouse). In this regard, we found that SQL is not suited for such analysis of categories, as recursion is required to determine related sub-/supercategories. Graph database systems may be a better fit for the category data. However, due to time constraints, we decided to generate a fixed list of pages from one category for analysis instead of importing data into a new system. Exemplary, we examined edits of articles in the ”2020 United States presidential election” category. We found that even on the day with the least amount of reverts in this category, the percentage of reverted edits was higher than the percentage of reverts across the whole English Wikipedia. This suggests that politics, particularly the U.S. election is a controversial and heavily discussed topic on the free encyclopedia.

C. Anonymous Edits

The core concept of Wikipedia is based on the free and easy collaboration of millions of voluntary authors worldwide. In order to promote knowledge sharing and lower the entry barrier for new contributors, editing Wikipedia does not require a user account. Most articles on Wikipedia can be edited anonymously with some exceptions for special, popular, and controversial articles. Anonymous edits are easily distinguishable as the editing user name is automatically filled with the respective author’s IP address.

Naturally, tracing specific users based on their IP address alone is impossible without internal knowledge of Internet service provider (ISP) network data. However, IP addresses can be easily mapped to their autonomous system (AS). As a result, we can determine the ISP affiliation of anonymous authors and their approximate geolocation. In this regard, we enrich edit events using geolocation data from the MaxMind GeoLite 2 database [19]. While IP geolocation databases generally offer questionable accuracy [20], approximate country-level locations suffice for our exemplary use case.

Table I provides an overview of observed anonymous edits for our time period. Interestingly, the percentage of anonymous edits on the English version of Wikipedia is significantly higher compared to the German counterpart. This is partly due to the fact that for the English Wikipedia, we observed a higher percentage of IP addresses with only one associated edit.

TABLE I
ANONYMOUS EDITS ON WIKIPEDIA

Wiki	Ratio	IPv4	IPv6	IPv4 Ratio
EN	15.3%	1.608.449	741.749	68.44%
DE	7.51%	114.902	77.301	59.78%

In addition, we see a higher IPv4 ratio for anonymous edits in the English Wikipedia. This generally suggests an increased IPv6 deployment in German regions. Unsurprisingly, a major share of English edits originates from English speaking countries, with the U.S, the U.K., India, Canada, and Australia among the top five. Similarly, we observed most edits in German originating from Germany, Austria, and Switzerland with Italy and France completing the top five.

Further, analysis of anonymous edits and their IP addresses may reveal interesting patterns and coherences. In the past, major media outlets reported several incidents where Wikipedia articles were purposefully edited to manipulate information. In particular, a report in 2006 [21] revealed that politicians had

asked their staff to make changes to their respective Wikipedia articles, e.g. replacing parts with a staff-written biography. An ensuing investigation [22] discovered multiple articles with similar extenuations as well as acts of vandalism.

We can confirm that these incidents are still happening regularly. For our time period, we captured and identified several anonymous edits originating from IP address ranges related to U.S. and German governmental institutions. In general, we consider this circumstance highly alarming. Especially with political topics, subjectivity is a particularly vital property for public encyclopedias. Naturally, the possibility of government officials editing Wikipedia articles to better suit their agendas and narratives embodies a major threat to freedom of speech and the stability of modern democracies.

V. Conclusion

In this work, we investigated how Wikipedia edit events can be observed, captured, processed, and analyzed in real-time in order to gain analytical insights and improve the understanding of the Wikipedia ecosystem. After revisiting the fundamental concepts of event stream processing and introducing our data source, we discussed related work demonstrating the relevance of Wikipedia edit metadata for analytical research. Subsequently, we designed and implemented a proof-of-concept event processing pipeline to streamline future analyses. In this regard, we proposed a Python-based EventStreams API client directing observed edits into separate topics on an Apache Kafka cluster. Leveraging Apache Kafka as a message queue and event streaming platform allowed us to facilitate processing and subsequent storage in a ClickHouse database. The analytical capabilities of the ClickHouse database enable real-time analysis and facilitate OLAP workloads.

Finally, we tested the viability of our proposed event processing pipeline by exploring potential analytical use cases. We found that a small number of users contributes a majority of edits, of which a significant share is executed by bots. Further, a small fraction of all edits is reverted, mostly to limit the spread of misinformation and prevent vandalism. Additionally, we experimented with the analysis of user interests by correlating article categories extracted from the Wikipedia category dump. By analyzing IP addresses of anonymous edits, we observed that a major percentage of edits originate in countries where the official language matches the language of the respective Wiki. Further, the English Wikipedia sees a higher edit rate of anonymous authors as well as a higher rate in IPv4 traffic compared to the German version.

While the proposed pipeline generally streamlines analytical research, it remains a general-purpose processing pipeline. For more specific use cases, the pipeline may be adapted for further performance improvements. Exemplary, Kafka streams could be used to build fine granular topics and preprocess data before ingestion into a database or analytical framework.

References

[1] Alexa. The top 500 sites on the web. [Online]. Available: https://www.alexa.com/topsites

[2] C. Wrench, F. Stahl, G. Di Fatta, V. Karthikeyan, and D. D. Nauck, "Data stream mining of event and complex event streams: A survey of existing and future technologies and applications in big data," in *Enterprise Big Data Engineering, Analytics, and Management*. IGI Global, 2016, pp. 24–47.

[3] R. Sahal, J. G. Breslin, and M. I. Ali, "Big data and stream processing platforms for Industry 4.0 requirements mapping for a predictive maintenance use case," *Journal of Manufacturing Systems*, vol. 54, pp. 138–151, 2020.

[4] C.-J. Su and S.-F. Huang, "Real-time big data analytics for hard disk drive predictive maintenance," *Computers & Electrical Engineering*, vol. 71, pp. 93–101, 2018.

[5] K. Teymourian, M. Rohde, and A. Paschke, "Knowledge-based processing of complex stock market events," in *Proceedings of the 15th International Conference on Extending Database Technology*, 2012, pp. 594–597.

[6] A. Acharya and N. S. Sidnal, "High Frequency Trading with Complex Event Processing," in *2016 IEEE 23rd International Conference on High Performance Computing Workshops (HiPCW)*. IEEE, 2016, pp. 39–42.

[7] Wikimedia Foundation. irc.wikimedia.org. [Online]. Available: https://wikitech.wikimedia.org/wiki/Irc.wikimedia.org

[8] ——. Event Platform/EventStreams. [Online]. Available: https://wikitech.wikimedia.org/wiki/Event_Platform/EventStreams

[9] W3C. Server-Sent Events. [Online]. Available: https://www.w3.org/TR/eventsource/

[10] T. Steiner, S. Van Hooland, and E. Summers, "MJ no more: Using concurrent wikipedia edit spikes with social network plausibility checks for breaking news detection," in *Proceedings of the 22nd International Conference on World Wide Web*, 2013, pp. 791–794.

[11] L. Lyu and B. Fetahu, "Real-time Event-based News Suggestion for Wikipedia Pages from News Streams," in *Companion Proceedings of the The Web Conference 2018*, 2018, pp. 1793–1799.

[12] Wikimedia Foundation. Manual:Pywikibot. [Online]. Available: https://www.mediawiki.org/wiki/Manual:Pywikibot

[13] ——. Help:Minor edit. [Online]. Available: https://en.wikipedia.org/wiki/Help:Minor_edit

[14] Apache Software Foundation. Apache Kafka. [Online]. Available: https://kafka.apache.org/

[15] Yandex. Distinctive Features of ClickHouse. [Online]. Available: https://clickhouse.yandex/docs/en/introduction/distinctive_features/

[16] Wikipedia. Wikipedia:Bots/Requests for approval. [Online]. Available: https://en.wikipedia.org/wiki/Wikipedia:Bots/Requests_for_approval

[17] ——. Wikipedia:Bot policy. [Online]. Available: https://en.wikipedia.org/wiki/Wikipedia:Bot_policy

[18] MediaWiki. Manual:categorylinks table. [Online]. Available: https://www.mediawiki.org/wiki/Manual:Categorylinks_table

[19] MaxMind. GeoLite2 Free Geolocation Data. [Online]. Available: https://dev.maxmind.com/geoip/geoip2/geolite2/

[20] I. Poese, S. Uhlig, M. A. Kaafar, B. Donnet, and B. Gueye, "IP geolocation databases: Unreliable?" *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 2, pp. 53–56, 2011.

[21] E. Lehmann. Rewriting history under the dome. [Online]. Available: https://www.lowellsun.com/2006/01/27/rewriting-history-under-the-dome-2/

[22] Wikipedia. Wikipedia:Congressional staffer edits. [Online]. Available: https://en.wikipedia.org/wiki/Wikipedia:Congressional_staffer_edits