BACHELORARBEIT / BACHELOR'S THESIS

Titel der Bachelorarbeit / Title of the Bachelor's Thesis

„Are the tides turning for microservices? Tracking the adoption of microservices in developer communities through sentiment analysis"

verfasst von / submitted by

Andreas Wallnöfer

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Bachelor of Science (BSc)

Wien, 2025 / Vienna, 2025

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	UA 033 526
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Bachelorstudium Wirtschaftsinformatik
Betreut von / Supervisor:	Dipl.-Ing. Patric Genfer
Mitbetreut von / Co-Supervisor:	Univ.-Prof. Dr. Uwe Zdun

Abstract

Microservice Architecture (MSA) has been a trending architectural style for years, with many companies migrating from traditional monoliths to more flexible microservices. However, MSA is not suitable for all use cases as it comes with its own drawbacks like network latency. That's why recently some companies have moved away from MSA. This raises the question of whether this trend in moving away from MSA can also be seen in how developers feel about MSA. Have their opinions toward MSA changed over time?

To answer this question, I performed an initial sentiment analysis of developers' discussions in online communities. After evaluating various online platforms as datasources, I selected Reddit and Hacker-News, because they contain many sentiment-filled discussions on MSA.

I developed a tool to extract MSA-related posts from these data sources, filter them, remove tags and special characters, and save each discussion as a CSV file, organized by year.

To perform sentiment analysis on this dataset, I researched existing tools and chose to test Senti4SD, EASTER, and ChatGPT to evaluate their ability to classify sentiments in developer comments.

I evaluated the accuracy of these tools by comparing their predictions with about 100 manually classified comments. This showed that the accuracy of these tools is not yet perfect. EASTER performed the best as a single tool, achieving a F1 score of 0.51. When considering only comments shorter than 50 words, EASTER's F1 score improved to 0.63. By combining the strengths of all three tools into a composite score, I achieved an F1 score of 0.66.

Finally, I analyzed a subset of my data to explore potential sentiment trends. The results show that negative sentiments increased from 17% in 2015–2017 to 25% in 2024, while neutral sentiments decreased from 76.67% to 67.7%. Positive sentiments remained stable at around 7%. These results suggest that developers indeed feel more negatively towards MSA now than they did in earlier years. However, given the subset size and tool limitations, these findings are not yet enough to draw conclusions.

Nonetheless, this trend makes future research into sentiment trends within online developer communities promising as there is a possibility that this negative trend could be confirmed.

Contents

1. Introduction

Microservice Architecture (MSA) has become a trending architectural style in software engineering. Initially, many companies shifted from monolithic systems to microservices due to advantages like flexibility and scalability. However, recently this trend seems to start reversing as some big companies like Amazon Prime Video are migrating from microservices back to monolithic architectures. [SLT23].

Developers' opinions on a technology can significantly affect their productivity and the quality of their work [LZB+18]. Understanding how developers feel about MSA is therefore interesting. Also researchers like Su et al. (2023) [SLT23] suggest the importance of analyzing developers' sentiments towards MSA as they concluded that more and more practitioners are seeing the benefits of switching back to monoliths.

While Bandeira et al. (2019) [BMPM19] explored microservice-related discussions on Stack Overflow, their study primarily focused on identifying discussion topics rather than examining developers' sentiments. This creates a clear research gap: analyzing how developers feel about MSA has not yet been investigated.

Lin et al. (2018) [LZB+18] emphasize the potential of automated sentiment analysis (SA) in software engineering research, where it has been applied successfully in different studies. This makes SA a promising approach for exploring developers' opinions about MSA.

This thesis analyzes whether the trend of moving away from microservices is noticeable in developers' sentiments. To address this, I analyze posts from online developer communities to understand how sentiments towards microservices have evolved over the years. The study represents a first step towards sentiment analysis of developer communities by collecting, filtering and preprocessing relevant posts, experimenting with various sentiment analysis tools and exploring initial sentiment trends in subsets of data.

The structure of this thesis is as follows:

- Chapter 2 reviews relevant literature on microservice architecture trends and MSA discussions on StackOverflow
- Chapter 3 provides the theoretical background on microservice architecture, sentiment analysis and statistical metrics which can be used to assess the accuraacy of tools
- Chapter 4 explores different developer communities as potential data sources
- Chapter 5 discusses the Extraction Tool I built for extracting posts and comments about MSA

- Chapter 6 compares different sentiment tools in literature and in my local setup. Then it explores whether a sentiment trend can be found in a subset of the data
- Chapter 7 concludes the thesis with a summary of key topics

To achieve these objectives, I have defined the following research questions.

1.1. Research Questions

RQ1: Which data sources are the most relevant to analyze the sentiment of developers towards microservice architecture?

RQ1 examines which data sources are suitable for extracting discussions about microservices to then analyze developer sentiment towards microservice architecture. Potentially interesting sources are GitHub [1], StackOverflow [2], Reddit [3], Twitter (X) [4], Dev.to [5], and Hacker News [6].

Not all of these sources offer sufficiently usable discussions or an accessible API, so it is necessary to find out which ones are most relevant and practical to use.

RQ2: Are there suitable tools for performing sentiment analysis on developer community content?

To answer this research question, a literature review of existing tools for sentiment analysis will be performed. Based on the findings, I will evaluate how accurately some of the promising tools can predict sentiment towards MSA compared to manual classification.

RQ3: Is there any indication in online communities suggesting a change in developers' emotions or sentiments over the years?

This research question looks at whether there are signs in online communities that developers' emotions or sentiments have shifted over time.

Research Questions Summary

To sum up, the main objectives of my bachelor's thesis are to identify the most relevant data sources for analyzing developer sentiment towards microservice architecture (RQ1), to evaluate the tools available for sentiment analysis (RQ2), and to analyze whether developers sentiments towards MSA expressed in online communities might have changed over the years (RQ3).

[1]`https://github.com`, Accessed: 2024-10-05.
[2]`https://stackoverflow.com`, Accessed: 2024-10-05.
[3]`https://www.reddit.com`, Accessed: 2024-10-05.
[4]`https://twitter.com`, Accessed: 2024-10-05.
[5]`https://dev.to`, Accessed: 2024-10-05.
[6]`https://news.ycombinator.com`, Accessed: 2024-10-05.

2. Related Work

2.1. Microservices as a Trend

Microservices have gained visible momentum over the last years as many companies like Amazon, Netflix, Google, IBM, Uber, and Alibaba switched to the microservice architecture according to Ren et al. (2018)[RWW$^+$18].

However, Su et al. (2023) [SLT23] discuss a new trend in which organizations are migrating from microservices to monolithic architectures (MA). Factors such as cost, complexity, scalability, performance, and organizational challenges have prompted this change.

The companies Segment (Customer data infrastructure), InVision (Digital product design platform), and Istio (Service mesh for microservices) were reported to have switched from MSA back to a monolithic architecture (MA). [SLT23]

The most notorious company that switched back from MSA to MA is the streaming platform Amazon Prime Video, which reduced its infrastructure costs by over 90% by moving away from microservices. [SLT23]

While the migration from Segment in 2020 didn't catch the eye of the public, Amazon's migration in 2023 did lead to heated discussions about the microservice architecture in articles and among developers.[SLT23]

Some practitioners called MSA "a zombie architecture" or said it is not a “utopian application architecture”.[SLT23]

It is important to note that the transition between different architectural styles is complex and requires serious effort. Therefore, a switch from monoliths to microservices or the other way around needs to be carefully planned, and before migrating one needs to be aware of the different strengths and weaknesses of the selected architecture.[SLT23]

2.2. Microservice discussions on StackOverflow

Bandeira et al. (2019) [BMPM19] explained that microservice architecture is very popular because of advantages like flexible horizontal scaling, which is crucial in cloud environments. This interest among developers is also seen in internet communities. That's why Bandeira et al. (2019) [BMPM19] decided to study microservice-related posts on StackOverflow to understand what topics developers were discussing.

As a data-source, they used 2,980 StackOverflow questions that were tagged with "microservices." The authors then filtered these discussions using five popularity metrics:

AnswerCount, ViewCount, CommentCount, FavoriteCount, and Score, extracting discussions that ranked above the third quartile for each metric. This led to a final corpus of 1,043 relevant discussions.[BMPM19]

The researchers used a natural language processing technique called "topic modeling" to analyze the data corpus for their underlying discussion topics. This allowed them to discover the underlying themes in the data and to find which topics were most discussed.[BMPM19]

Their results show that "Token Authentication" and "Message Queues" were the most discussed topics in both technical and conceptual discussions. The topic "containers", was mostly talked about in technical discussions. One major topic in conceptual discussions was the difference between "SOA" (service-oriented architecture) and "microservices", which made up around 10% of the conceptual discussions [BMPM19].

Also notable was that about 13% of the technical posts about microservices focused on the technology "Netflix Eureka" [BMPM19].

The study also found that just because a topic had many views, that doesn't mean that it is the most popular technology among developers. For example, they noticed that Microsoft technologies were discussed more often than Amazon technologies, even though Amazon had a larger market share at the same time [BMPM19].

3. Theoretical Background

3.1. Microservice Architecture

Microservice architecture (MSA) is a software architectural style that splits an application into small, independent services, [AMPJ17] which are loosely coupled and follow the principle of single responsibility. [PZZ$^+$22]

Microservice Architecture (MSA) is based on similar ideas as Service-Oriented Architecture (SOA), with the key difference being their focus. SOA is about integrating systems across an entire enterprise, whereas MSA is used mostly for large-scale, cloud-based applications.[GKG22]

A microservice can be reused by multiple applications if made easily accessible through well-defined ways like REST-APIs .[SA24]The communication between different components happens with lightweight protocols.[AMPJ17] like HTTP, REST, RPC, and messaging patterns.[PZZ$^+$22]

Additionally, each service operates within its own process and can be tested, scaled and deployed independently.[AMPJ17] Automated DevOps pipelines with tools like Kubernetes are often used for the deployment. [KSNNSN24]

Monoliths

The monolithic architecture (MA) is one of the most prominent alternatives to MSA, as it used to be one of the more dominant styles in software architecture.

Monoliths are designed as a single unit. Therefore, they tend to have tight coupling between all parts and have to be developed, packaged, and deployed as a whole. In contrast to Microservices, Monoliths don't need to communicate with parts of their system through a network and can even use a single database. [Ham23, GKG22].

Historically, monoliths were the standard architecture, aligning well with development methodologies like waterfall and the needs of pre-cloud infrastructure [GKG22]. However, in the cloud computing era more flexible architectures that are also able to use Infrastructure as a Service (IaaS) dynamically might be better.[GKG22].

Another significant drawback of monolithic architecture (MA) is the inability to scale components individually, requiring the entire application to be rebuilt and redeployed for any code changes.[RWW$^+$18, GKG22].

Additionally, MA is bounded to a single tech stack, hindering the adoption of new technologies and making the use of legacy applications harder, as some technologies might be outdated or not well known among developers anymore. [RWW$^+$18]

However, some of the drawbacks people associate with monolithic architectures (MA) might actually be due to bad design choices rather than the architecture itself. A well-designed MA can still be a solid choice in some situations. For instance, MA helps ensure data consistency and can make debugging easier.[GKG22].

The choice between MSA and MA should be heavily influenced by the usecase. A well-designed monolith that respects best practices such as data encapsulation and modularity can be the right choice for organizations with limited resources [GKG22], while MSA might be better for larger organizations with diverse and rapidly changing requirements [PZZ+22].

Advantages of MSA

Microservice architecture (MSA) offers significant strengths, including scalability, maintainability, and reusability from breaking down monolithic applications into independently deployable services.[KSNNSN24] The failure of one service doesn't bring down the entire system, meaning MSA makes the software more fault tolerant and easier to mantain.[Ham23]

Furthermore, the software industry is shifting from traditional project management approaches, such as the waterfall model, to more agile methodologies like Scrum, XP, and Kanban. These agile methods focus on flexibility and the ability to quickly adapt to changes, which align well with the strengths of MSA.[GKG22]

Moreover, the modularity of microservices allows different teams to develop and maintain components independently and to use diverse technology stacks and repositories. [PZZ+22]. So one microservice can be developed in Java and another one in Python, as long as they are able to communicate over a common interface like a Rest API.

When applications need to work concurrently and support many users at once, a Microservice Architecture is effective because it breaks the app into smaller parts, making it easier to scale and manage.[RWW+18]

The advantages of MSA lead many organizations to migrating from MA to MSA.[Ham23] However, MSA isn't fitting for all usecases as it also comes with disadvantages.

The drawbacks of MSA

A new trend is emerging currently, where companies are switching back from MSA to monoliths, as microservices come with their own problems.[SL24]

The companies Istio (Service mesh for microservices) , Segment (Customer data infrastructure) and Amazon Prime Video (Streaming Platform) reported high operational costs due to MSA, which led them to switch back to monoliths. [SLT23]

Also the company InVision (Digital product design platform) claimed that MSA can be expensive because they need to keep backup systems to ensure everything runs smoothly and remains available, even if one part fails. [SLT23]

Another issue appears when the number of microservices grows too high. This introduces complexity and increases potential reliability issues due to dangers like network errors and sudden load peaks. [Wan19].

The company Istio uses different tech stacks for different microservices, which in general can lead to high flexibility, but in their case they reported that this complicated management, as it led to having multiple teams with different technologies. Segment and InVision had similar complaints and struggled with the management and maintenance of multiple repositories, databases, tech stacks, tools and libraries .[SLT23] On the other hand, both the companies Appsmith (Low-code internal tools) and Shopify (E-commerce platform provider) praised monoliths for the benefit of a single codebase with clear component boundaries, as this enables them to deploy a unified binary. [SL24]

Another major challenge is scalability. Although microservices architecture is generally considered scalable, it can face problems such as complex interactions, managing resources, network latency and deployment issues.[SLT23]

Also performance issues were reported. One major problem Segment reported is head-of-line blocking, where slow requests delay all services, decreasing response times. [SLT23] For the companies Shopify (E-commerce platform provider), Appsmith (Low-code internal tools), Gusto(Payroll and HR software), and PlayTech (Online casino software) performance was a factor in the decision to choose modular monoliths over microservices and Amazon Prime Video reported that the passing of video frames between components is inefficient and costly.[SL24]

Gusto decided not to use microservices because they bring their own challenges in testing and deployment. Plus, if they set the service boundaries wrong, fixing it would be really expensive.[SL24] Playtech too said maintenance of MSA would be unnecessary overhead. [SL24]

Summary

There has been a strong trend of companies migrating from MA to MSA due to the benefits of MSA and the rise of cloud computing and agile methodologies [GKG22].

However, recent developments show that some organizations, including the streaming platform Amazon Prime Video, have switched back to MA, lighting up discussions about the suitability of MSA for certain use cases [SLT23, SL24]. This shift indicates that while MSA has its advantages, it is not a universal solution, and the choice between MSA and MA should be carefully considered based on specific organizational needs and product requirements [GKG22].

3.2. Sentiment Analysis

Sentiment Analysis (SA) is a technique to automatically classify if a text contains positive, negative or neutral opinions. For this task Natural Language Processing can be used.[BKPVS20]

The sentiment analysis tool-set can be used for different tasks. For instance, it can be used for hate speech detection, which identifies language that spreads hate or discrimination against specific groups or individuals.

It can also be used for sentiment classification, where the goal is to categorize text based on its emotional tone as positive, negative, or neutral, which is the usage that is interesting for this Bachelor thesis[ZITL23].

Sentiment analysis is applied across various fields, including business, marketing, politics, healthcare, and public advocacy [ZITL23]. Sentiment analysis has also been used in the Software Engineering domain to understand developers emotions, where platforms such as the issue tracker Jira [1] and the repository management platform Github [2] have been used to analyze developers' sentiments in order to improve software engineering processes[NGL18] and improve developer productivity and task outcomes.[LZB+18]

Sentiment Analysis (SA) in the Sotware Engineering (SE) domain

Software Engineering discussions on online platforms can help to understand if developers are satisfied with a technology. That's why researchers like Sun. et. al. [SSG+22] explore the usage of sentiment analysis (SA) tools in the software engineering (SE) domain.

Off the shelf SA tools that work well for social media posts have been shown to not produce accurate results for the SE domain. [ZITL23] These tools have been trained on texts unrelated to software engineering, like movie or product reviews, which means they may incorrectly identify the sentiment polarity in software engineering related texts. [JDS15] For instance, the word "robust" in a movie review does not indicate a particular emotion, but when it refers to a software product it has a clear positive meaning.[LZB+18]

Another complication is that developers' topics are often very specific, requiring them to describe their issues before expressing their sentiments. This leads to more complex sentence structures than those found in simple movie reviews, which often confuses off-the-shelf tools and makes sentiment analysis difficult in the software engineering domain.[SSG+22]

To address these shortcomings, different researchers have begun developing their own SE-specific tools.[LZB+18][SSG+22]

Some tools are created by adjusting existing models with a software engineering dictionary, while others retrain machine learning solutions on SE data.[SSG+22]

While customization improves performance, it does not automatically make the tool 100% accurate. It's important to keep in mind that different tools can have different accuracy depending on the data source. For example, a tool trained on Stack Overflow data may not reach the same results when applied to JIRA data. [LZB+18] Therefore B. Lin et al. (2018) [LZB+18] and Sun et al. [SSG+22] highlight how important it is to evaluate tools based on specific domains and to make sure the tools can work well in different domains.

[1] `https://www.atlassian.com/de/software/jira`

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

Data preparation

To be able to use sentiment analysis tools on texts from online platforms usually some preprocessing steps have to be done.[IZ17] This can include removing elements such as code segments, URLs, special characters, and HTML tags, that do not have any emotional meaning but might mislead a NLP tool.[BKPVS20]

For lexicon-based tools, preprocessing often includes tokenization, stop-word removal, and lemmatization. [GH23]

Tokenization involves splitting the text up into individual words or phrases. [GH23]

Stop word removal gets rid of common words, such as "of" "for" or "the", that don't have any emotional meaning and create an extra computational load, potentially leading to errors if not removed.[GH23]

Lemmatization is another possible preprocessing step, where words are reduced to their root form (for example, "running" becomes "run"), making it easier for lexicon-based tools to compare the word to the dictionary and analyze the contained sentiment.[GH23]

Typos can also mislead some tools, for instance, Sentistrength-SE tries to correct typos automatically to enhance the results. [IZ17].

The preprocessing steps needed are highly dependent on the sentiment tools. For instance, Sentistrength-SE, Senti4SD and EASTER take over some preprocessing steps like Tokenization. [SC24, CLMN18, SSG$^+$22] Furthermore, the tool "EASTER" [SSG$^+$22] requires as input a CSV file with a semicolon (;) as the delimiter, so the text must not contain any semicolons, as this causes exceptions for the tool, which thinks all semicolons are delimiters.

Sentiment Scale

Sentences can contain emotions, which can be quantified with the measurements polarity and intensity. [IZ17]

Sentiment polarity captures if an expression contains a positive, negative, or neutral emotion. [IZ17]

Sentiment intensity measures the level or strength of the emotion. Many sentiment analysis tools measure these aspects through numerical scores, but the sentiment scales vary for different tools. [IZ17]

The SentiStrength scale employs two pairs of integer values, with positive scores ranging from $+1$ to $+5$ and negative scores ranging from -1 to -5 and 0 for neutral. [SC24]

In contrast, other tools like EASTER only differ between negative (-1), neutral (0) and positive (1) sentiment. [SSG$^+$22]

Workflow of a SA tool

There are different approaches to constructing sentiment analysis tools, which each have their strengths and weaknesses. I will explore some different tools in Section 6 and want to explain the lexicon based approach as a simple example now.

Lexicon-based sentiment analysis tools like SentiStrength-SE evaluate sentiment by assigning positive and negative scores to individual words and summing them. They rely on a dictionary, where words have preassigned sentiment scores. [IZ17]

Lexicon-based tools can run into issues because they don't consider context within a sentence and focus only on the individual words. For instance, a sentence like "I would not recommend this, even though it is robust and fast" could be misclassified as positive due to the presence of positive words like "robust" and "fast.", even though the "I would not recommend" part of the sentence clearly shows a negative sentiment. [IZ17]

This issue is handled better by more complex tools like EASTER, that take context into consideration by working with neural networks and deep learning techniques.[SSG+22]

Different approaches have different strengths and weaknesses. In section 6 I will make a short comparison of the most common sentiment analysis tools in software engineering research.

3.3. Metrics for Accuracy Calculation

There are different metrics, which could be used to evaluate how accurately a tool's predictions reflects the ground truth. I use the statistical metrics Precision, Recall and F1-score to evaluate the accuracy of sentiment anaylsis tools. This was inspired by the research on EASTER [SSG+22], where the researchers used the same statistical values.

Here is what these metrics mean:

- **Precision (P)**: Measures the proportion of items classified as positive that are actually positive according to the labels in the input file, or the same for neutral and negative, therefore giving the accuracy of positive/neutral/negative predictions.

$$P = \frac{\text{Number of correctly classified texts with sentiment } c}{\text{Total number of texts classified as sentiment } c}$$

- **Recall (R)**: Checks how many actual positives the model identified correctly as positive or the same for neutral/negative, showing the effectiveness of finding the right sentiment polarity

$$R = \frac{\text{Number of correctly classified texts with sentiment } c}{\text{Total number of texts that actually have sentiment } c}$$

- **F1-measure (F1)**: Combines Precision and Recall into a single score

$$F1 = \frac{2 \times P \times R}{P + R}$$

[SSG+22]

4. Developer Community Analysis

Companies analyze social media comments from platforms like Reddit [3] to find out about user needs. [Sha21]

Researchers are also using the analysis of comments from social platforms to get insights about different research topics. It's useful that such platforms can provide historical comments and therefore make it possible to analyze the changing of a phenomenon over time. Overall, social news websites like Reddit [3] and HackerNews[6] can strengthen the validity of ideas in research.[BJMH15]

Therefore I use social platforms as my data sources for this study. I compare GitHub [1], StackOverflow [2], Reddit [3], Twitter (X) [4], Dev.to [5], and HackerNews[6] as potential data sources. This chapter investigates RQ1. 1.1

4.1. Community Comparison

I evaluated several online communities, including GitHub [1], StackOverflow [2], Reddit [3], Twitter (X) [4], Dev.to [5], and HackerNews[6] as potential data sources. I also explored Discord [1] and Google Groups, such as "groups.google.com/g/microservices". However, Discord discussions are private, and Google Groups contain too much spam, which eliminated them for me.

My primary goal in the community comparison was to identify a platform that provides access to data through a publicly available API while also offering enough content in the form of emotional discussions related to Microservices Architecture (MSA). The results of my evaluation are summarized in Table 4.1, and in the following bullet points:

- GitHub[1]: There were only limited discussions about the concept of MSA available, the bigger focus was on technical issues. The discussions did mostly have not enough comments and overall there weren't enough opinions about the MSA concept on GitHub. That's why I eliminated this data source.

- StackOverflow[2]: This platform primarily featured technical questions and lacked conceptual discussions about MSA, making it unsuitable for my research needs.

[1]`https://github.com`, Accessed: 2024-10-15.
[2]`https://stackoverflow.com`, Accessed: 2024-10-15.
[3]`https://www.reddit.com`, Accessed: 2024-10-15.
[4]`https://twitter.com`, Accessed: 2024-10-15.
[5]`https://dev.to`, Accessed: 2024-10-15.
[6]`https://news.ycombinator.com`, Accessed: 2024-10-05.
[1]https://discord.com/

Furthermore, most posts were neutral as this is what StackOverflow wants from its users as a platform. But this isn't good for Sentiment Analysis and therefore StackOverflow was eliminated from my research.

- Reddit [3].: Reddit provides numerous threads with relevant discussions about MSA, including emotional content, making it a promising source for my research. The API is free, but it's necessary to use a access token that can be generated without costs. I decided to look further into Reddit as a data source.
- Twitter (X)[4]: The free API only offers writing endpoints and the reading API starts at $100 per month. Moreover, the threads I found did not contain enough valuable discussions about MSA.
- Dev.to[5]: This community mainly consists of blog articles with few relevant comments regarding MSA, therefore not meeting my requirements.
- Hacker News[6]: This platform contained many relevant discussions with emotional comments, proving to be a valuable resource. The API is freely usable. I decided to look further into HackerNews.

Based on my findings, I'm able to answer RQ1 1.1. Reddit [3] and Hacker News[6] are the most suitable datasources for my study and I choose them as my primary sources for data collection regarding emotional discussions about MSA.

4.2. Further Analysis on Reddit & HackerNews

Reddit and Hacker News have some key similarities as platforms. Both use a transparent, non-personalized interface that shows all users the links in a very similar ranking. [Sto15]

Both rely on user votes to calculate post popularity, which makes the ranking process transparent. It's interesting to note that two articles of similar quality can receive very different levels of attention, reflecting the "rich get richer" phenomenon where popular posts get even more popular. Despite this, studies show that the most popular content on these 2 platforms is generally of higher quality compared to less popular posts. [Sto15]

Reddit

Reddit is a platform with over a million specialized communities, known as subreddits. [Sha21] Each subreddit or short "sub" typically focuses on a specific topic, such as "r/java" or "r/wien". This gives users a space where they can find people with similar interests and engage in discussions.[Sha21]

Inside a subreddit, the posts are ranked according to upvotes, downvotes, and the age of the article. [Sto15]

[7]`https://support.reddithelp.com/hc/en-us/articles/16160319875092-Reddit-Data-API-Wiki`, Accessed: 2024-10-05.

Posts on Reddit are also categorized with tags in the metadata, which are called flairs. Flairs help to identify themes, such as humor, within the content. [Sha21]

Noticeable on Reddit is how often posts get reposted. [Sto15] This is not harmful to my research since I'm more focused on the comments rather than the original articles or media.

The Reddit rules [7] allow the publishing of research results if the Reddit data and its derivatives like models trained on Reddit data are excluded. It's also mandatory to credit Reddit, anonymize information, and provide a copy of the research to "reddit.research@reddit.com" before publishing. The rate limit is 100 queries per minute.[7]

HackerNews

HackerNews(HN) is a social news platform that attracts mainly a technical audience. It has strict moderation guidelines, which helps enhance the quality of the comments.[BJMH15]

Overall I found 30.531.421 comments on HackerNews by using an empty comment search on hn.algolia [9].

The rate limits of the hn.algolia API[9] from a single IP are 10.000 requests per hour. HackerNews ranks the stories by points divided by the power of the time since a story was submitted.[BJMH15]

The posts on Hacker News can be categorized into the following types: Jobs, Polls, Ask HN, Show HN, stories, and comments.

The Show HN category is primarily intended for users to share their personal projects, making it irrelevant for my research focus.

Jobs posts are not interesting to my study.

Polls only had 1 result in a search [8] for "microservice" and is therefore irrelevant as well.

Stories consist of news articles that generate extensive discussions in the comments, which will be valuable for my research. Therefore, I will focus on analyzing the stories and their associated comments.

The AskHN section contains questions and other text submissions, asking the HN community for advice. However, I had fewer results for a search for "microservice" in AskHN (493 results [8]) than in stories for the same search (4453 results [8]) and found less interesting discussions in AskHN compared to the stories category.

[8]`https://news.ycombinator.com`, Accessed: 2024-10-05.
[9]`https://hn.algolia.com/`, Accessed: 2024-10-05.

Platform	Free API Available	Discussions	Conclusion
GitHub[1]	Yes	Not enough useful discussions found about MSA.	Not relevant, too few discussions.
StackOverflow[2]	Yes	Mostly technical questions, few conceptual discussions about MSA. Neutral and informative.	Not relevant, not enough conceptual discussions containing opinions.
Reddit [3]	Yes (API key required)	Many threads with relevant discussions about MSA, including emotional content.	Relevant.
Twitter (X)[4]	Not free (Starts at $100/month)	Most replies under MSA posts are not relevant and not enough comprehensive discussions were found.	Not relevant, not enough relevant data, and API costly.
Dev.to[5]	Yes	Mainly blog articles, few relevant comments about MSA.	Not relevant, not enough comprehensive discussions were found.
Hacker News[6]	Yes	Many relevant discussions with emotional comments.	Relevant.

Table 4.1.: Comparison of Developer Community Platforms

5. Dataset Extraction Tool

My dataset extraction tool extracts microservice related posts from Reddit and HackerNews, filters them, preprocesses the content and saves them to CSV files. The Tool was developed in Python and can be found on Github[1]. The main process of how the tool works is shown in Figure 5.1.

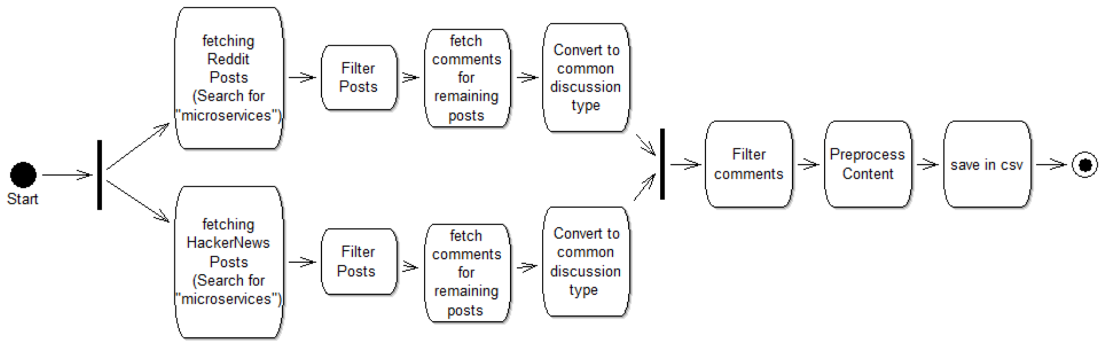
Start
fetching Reddit Posts (Search for "microservices")
Filter Posts
fetch comments for remaining posts
Convert to common discussion type
fetching HackerNews Posts (Search for "microservices")
Filter Posts
fetch comments for remaining posts
Convert to common discussion type
Filter comments
Preprocess Content
save in csv

Figure 5.1.: UML Activity Diagram for the Main Process of the Extraction Tool

I will go into more detail about the phases in the following subsections.

5.1. Data Extraction

This subsection describes how I fetch microservice posts from Reddit and Hackernews.

5.1.1. Reddit

I accessed the Reddit API using an Access Token generated on the Reddit API website[2]. You can find a short description on how to get a Access Token in the README of the Extraction Tool. [3]

To interact with Reddit, I use the Python library PRAW (Python Reddit API Wrapper [4]).

[1]https://github.com/andreasw21/microserviceAnalysis/tree/master/Extraction_Tool
[2]https://www.reddit.com/r/reddit.com/wiki/api/
[3]https://github.com/andreasw21/microserviceAnalysis/tree/master/Extraction_Tool
[4]https://praw.readthedocs.io/en/stable/index.html

The generated Access Token was integrated into PRAW to authorize API requests. This is done by initializing PRAW with the necessary authentication information like shown in 5.1.

```
import praw

class RedditAccess:
    accessor = praw.Reddit(
        client_id=my_client_id,
        client_secret=my_client_secret,
        user_agent="MicroserviceRelatedSubRetriever/1.0 by u/Andreas-W-
    research",
    )
```

Listing 5.1: Initialization of Reddit Access

To retrieve Reddit posts on a specific topic, I fetched all posts containing the search term "microservice", which were 244 posts. I utilized the search function of the PRAW library, which allows searching for posts across various subreddits and sorting them by relevance as seen in 5.2.

```
reddit = RedditAccess.accessor
submissions = reddit.subreddit("all").search(
    search_term, sort="relevance", time_filter="all", limit=num_posts
)
```

Listing 5.2: Search Query for Post Retrieval

The retrieved submissions contain various metadata, which is detailed in the official PRAW documentation[5]. For my analysis, the following metadata is of particular interest, which I temporarily store in a custom class called `RedditPost`:

- `id (str)`: The unique ID of the Reddit post.
- `title (str)`: The title of the Reddit post.
- `score (int)`: The score (sum of upvotes/downvotes) of the post.
- `created (date)`: The creation date of the post.
- `subreddit (str)`: The subreddit where the post was submitted.
- `content (str)`: The text of the post (if it is a text post) or the URL (if it is a link post).
- `flairs (List[str])`: A list of flairs associated with the post.
- `num_comments (int)`: The number of comments under the post.

[5] `https://praw.readthedocs.io/en/stable/code_overview/models/submission.html`

These structured `RedditPost` instances will be filtered as explained later in 5.2.1.

After filtering, I fetch the comments for the remaining posts using the PRAW library again. I use post IDs to get the submissions. Then, I grab all the comments. To make sure I don't miss any and get all the comments, not just the first page, I use the replace_more method. The code in 5.3 shows how a simple comment fetching algorithm in Python could look like.

```
import praw
submission = RedditAccess.accessor.submission(post.id)
submission.comments.replace_more(limit=None) # replace pagination objects
    with real comment
comments = submission.comments.list()  #  get all comments
```

Listing 5.3: Fetch Reddit Comments with Praw

5.1.2. Hacker News

For Hacker News (HN), no access token is needed. I can retrieve the so-called "stories," using the base URL[6]. Stories are links to news articles under which comments can be found. For the search for "microservice" I had 4.614 results.

A request to the algolia url[7] gives a JSON response that includes a "hits" array, where each "hit" represents a story. Each hit contains various metadata, detailed in the official API documentation[8].

For my analysis, I focused on the following metadata, which I saved into a custom class called `HN_Story`:

- `id (str)`: The unique ID of the Hacker News story.
- `title (str)`: The title of the Hacker News story.
- `score (int)`: The points (upvotes) of the Hacker News story.
- `created (date)`: The creation date of the Hacker News story.
- `content (str)`: The content of the Hacker News story, or the URL to the article.
- `num_comments (int)`: The number of comments under the story.

These structured `HN_Story` instances are then filtered as explained later in 5.2.1.

Fot the remaining posts I fetch the comments by again sending a get request to the algolia url[9] but this time including the story id for which we want the comments and setting the tag "comment". The Code in 5.4 shows a simple Python structure of how this could be done.

[6] http://hn.algolia.com/api/v1/search
[7] http://hn.algolia.com/api/v1/search
[8] https://hn.algolia.com/api
[9] http://hn.algolia.com/api/v1/search

```
url = f"https://hn.algolia.com/api/v1/search_by_date?tags=comment,story_{
    story_id}&hitsPerPage={all_hits}"

    comments = requests.get(url).json().get("hits", [])
```

Listing 5.4: Fetch HackerNews Comments with Algolia

5.2. Data Filtering

Directly utilizing all results from a search for "microservice" is not effective for my use-case, because many of these results contain technical posts, tutorials explaining the implementation of microservices, or discuss specific issues users had with a MSA technology, which is not interesting for my analysis. I'm looking for posts that discuss the concept of microservices architecture (MSA) and include comments that express emotions related to MSA.

That's why I apply multiple filtering steps to refine the dataset and isolate the relevant discussions. The filtering strategy I created for this study is based on my practical experience. I searched for the term "microservice" on both Reddit and Hacker News, looked at the results, and tried to figure out how to identify potentially relevant discussions. Based on these insights, I developed a series of heuristic filtering steps.

5.2.1. Posts Filter

Before even fetching the comments belonging to the original posts, I filtered the posts by their attributes like title and score. This filtering includes the following steps, which can be seen on the diagram 5.2.

- **Initial Data**: I began by searching for posts containing the term "microservice" on both HackerNews and Reddit. This is the initial data before applying the filtering steps
- **Relevance of Story Domain**: For HackerNews, the posts usually contain links to articles. A post's domain is analyzed to find out its relevance. If a link directs to GitHub, the post is considered irrelevant, as it likely talks about a coding issue rather than a conceptual discussion. The same applies to YouTube, which often features "funny" content.
- **Score Threshold**: I set a threshold based on the third quartile of post scores and filtered out the posts that fall below this threshold. This idea was inspired by a study that extracted microservice-related posts from StackOverflow and also took the third quartile for each metric to extract the most relevant discussions.[BMPM19]

  Both Reddit and HackerNews provide a score with their metadata, but they differ slightly:

  - **Reddit**: Scores are derived from upvotes and downvotes.

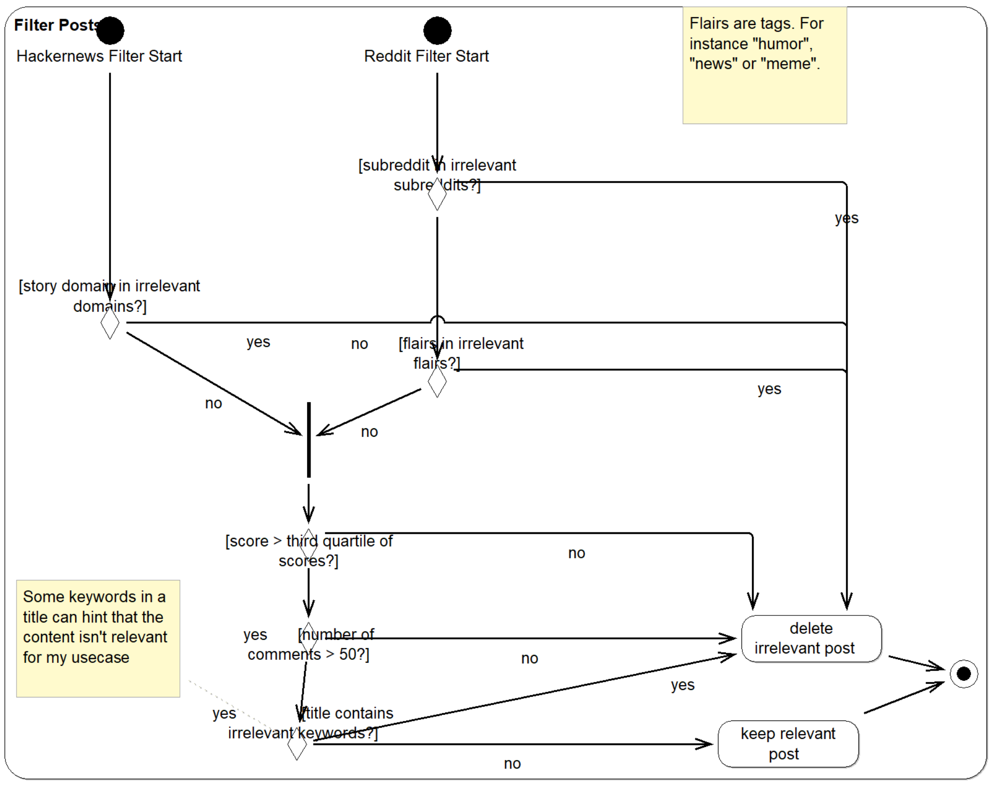
Filter Posts
Hackernews Filter Start
Reddit Filter Start
Flairs are tags. For instance "humor", "news" or "meme".
[subreddit in irrelevant subreddits?]
yes
[story domain in irrelevant domains?]
yes
no
[flairs in irrelevant flairs?]
yes
no
no
[score > third quartile of scores?]
no
Some keywords in a title can hint that the content isn't relevant for my usecase
yes
[number of comments > 50?]
no
delete irrelevant post
yes
yes
[title contains irrelevant keywords?]
keep relevant post
no

Figure 5.2.: UML Activity Diagram showing the Post filtering

  - **HackerNews**: Only upvotes are counted.

- **Number of comments**: I focused on discussions with more than 50 comments, as I considered this a reasonable threshold to ensure that there were enough comments for a meaningful discussion.

- **Title contains irrelevant Keywords**: From my experience, I think the title can already hint at whether a thread is a relevant discussion. That's why I decided to do a keyword-based filtering on the title.

  I developed a heuristic keyword list based on my experience. Posts containing certain company names, such as "Netflix," in their titles are in my experience likely to be general news stories rather than conceptual discussions focused on microservices architecture. Additionally, technical terms like "Spring" indicate implementation discussions rather than conceptual ones. This heuristical keyword list can be found in [10]

- **Relevance of Subreddit**: The subreddit where a Reddit post is located is important for filtering the results. Some subreddits mainly focus on humor or unrelated topics.

  I created a heuristic list of subreddits that seem irrelevant for my usecase. To do that I analyzed which subreddits my extracted posts belong to. Then I manually reviewed a few MSA posts for each subreddit to see if they contain interesting conceptual discussions about MSA or rather discuss technical questions or specific bugs. This helped me come up with a list of subreddits that I consider irrelevant for my research:

```
IRRELEVANT_SUBREDDITS = ["dotnet", "programmerhumor", "sysadmin",
                         "cscareerquestions", "de_edv", "nintendo",
                         "aws", "django", "videos", "node",
                         "postgresql", "springboot", "sre", "cpp",
                         "moonlightstreaming", "corporatefacepalm",
                         "supabase", "kubernetes", "clojure",
                         "apachekafka", "voip", "rails",
                         "opentelemetry", "azure", "gradle"]
```

- **Relevance of Flairs**: Flairs are tags associated with Reddit posts that provide insight into the nature of the discussion and can be used to determine whether a discussion is relevant for my use case.

  Similarly as for subreddits, I now analyzed which flairs my extracted posts have and manually reviewed if posts with a specific flair tend to be irrelevant. During

[10] https://github.com/andreasw21/microserviceAnalysis/blob/master/Extraction_Tool/src/Filter/keywords.py

the analysis, I noticed that many posts did not have any flairs and some flairs like "meme" made it very clear that the posts were not usable for my use case.

Here are the flairs that I consider irrelevant for my use case:

```
IRRELEVANT_FLAIRS = [  "meme","humor","schizoposting","tutorial","
intermediate showcase","cloud firestore","carreira",]
```

All posts containing these flairs are excluded from my results.

5.2.2. Comments Filter

After filtering the posts I fetched the comments for the posts. This happened in a similar way as the post extraction described in .

Not all comments in a discussion are useful for my use case. Some comments may confuse the sentiment analysis tool, while others might not even talk about microservices.

Therefore I created a heuristic filtering process for my comments that can be seen in Figure 5.3 and has these steps:

- **Deleted Comments**: If a comment is marked as deleted by having "[deleted]" or "[removed]" as its content, it is removed from the dataset.
- **Comment Length**: Next, I evaluate the length of each comment. Comments shorter than 4 words are excluded from the dataset. This is because brief comments tend to be less meaningful in my experience, often consisting of phrases like "Good idea" or "Interesting approach," and either comment directly on what the original post said thus not directly expressing an opinion about MSA or don't make it clear if they are actually talking about MSA.
- **Keyword Relevance**: Finally, I check whether the comment contains the keyword "microservice." Comments that include this keyword are kept, while those that do not are removed. This step tries to ensure that the comments are really about MSA, removing noise from the dataset.

5.3. Data Format

I define a discussion as a post with its associated comments. A post can contain text written by the author or a URL to a news article or media as its content.

I am interested in the creation date of both the post and its comments so that I can analyze potential relationships between sentiment and time.

I also save the source (Reddit or HackerNews) and the comments' IDs assigned by each platform, which allows me to track the comments easily.

The sentiment for each comment is initialized with a default value of 0, which can later be replaced using a sentiment analysis tool.

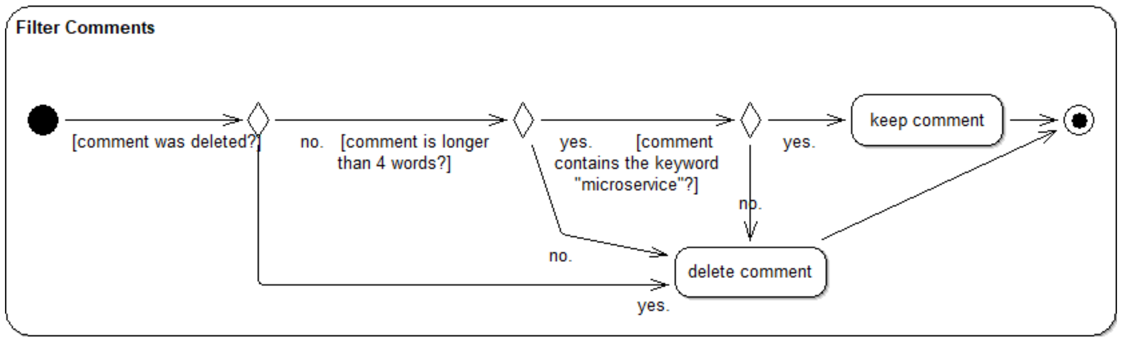

Figure 5.3.: UML activity diagram for the comment filter process

These discussions are saved in separate CSV files to maintain a clear overview of the data. Each discussion is stored in its own file, organized into folders by year (e.g., `Discussions/2024`, `Discussions/2023`). The naming pattern is Source-ID.csv, for instance "Reddit-1dmi48v.csv".

Each CSV file is structured as follows:

- **ID**: The unique identifier for each post or comment.
- **Source**: Indicates whether the data originates from Reddit or HackerNews.
- **Creation Date**: The date when the post or comment was created.
- **Content**: The text of the post or the URL to the article.
- **Sentiment**: A placeholder for the sentiment score of the comment with a default of 0.

I utilize two main classes to represent the data: `Comment` and `Discussion`. A discussion contains multiple comments. This is also reflected in the CSV, where we first have the original posts and then in the same file all the comments.

Their attributes are similar to those in the CSV and can be seen in Figure 5.4.

5.4. Data Preprocessing

Before the data can be analyzed by a sentiment analysis tool, it is useful to perform some preprocessing, as explained in Section 3.2.

To visualize the preprocessing process, I created a UML activity diagram, which is shown in Figure 5.5.

I performed the following preprocessing steps to ensure the quality of the comment data:

- **Single Line Transformation**: The discussion text is condensed into a single line. If it contained multiple lines, the resulting CSV would be confusing and unusable.

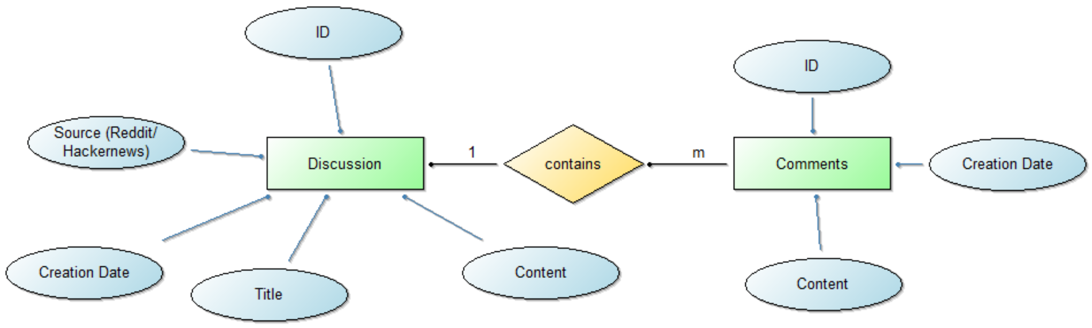

Figure 5.4.: Discussion with Comments

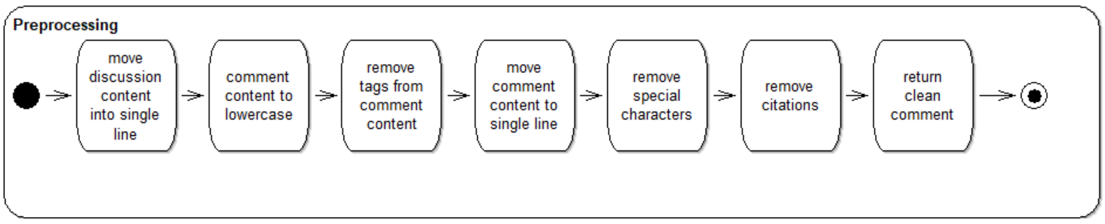
Preprocessing
move discussion content into single line
comment content to lowercase
remove tags from comment content
move comment content to single line
remove special characters
remove citations
return clean comment

Figure 5.5.: UML Activity Diagram for the Filtering Process (Top-to-Bottom Layout)

- **Lowercase Conversion**: The content is converted to lowercase. This is needed by some tools and doesn't seem to hurt the other tools.
- **Tag Removal**: Any tags present in the comment contents are eliminated, as they do not contain sentiment or context and might confuse the tools.
- **Special Character Stripping**: Special characters are removed from the text. This is needed for some tools and also it makes it possible to use delimiters like ";" in the CSV, that otherwise might be contained in the text.
- **Quote Removal**: Quotes that are marked with "<i>" tags are removed from the comments so that we don't evaluate the sentiment in the same sentence multiple times.

The preprocessed discussions are then saved into a Discussions Folder, in the subfolder which represents their creation date. Each discussion is in each own CSV file.

5.5. Setup of the Extraction Tool

To setup the Extraction tool you have to clone the tool [11]. Then you have to get a Reddit Access token and put it into the "Extraction_Tool/src/RedditApi/reddit_user_config.py"

[11] `https://github.com/andreasw21/microserviceAnalysis/tree/master/Extraction_Tool`

file like explained in the README[12]. The project was developed with Python version 3.12.4. All necessary dependencies can be installed by running "pip install -r requirements.txt".[13]

To run the data extraction, navigate to the ExtractionTool directory and use the command py -m src.main to extract posts and save them into a Discussions folder.

[12]https://github.com/andreasw21/microserviceAnalysis/blob/master/Extraction_Tool/README.md

[13]https://github.com/andreasw21/microserviceAnalysis/blob/master/Extraction_Tool/requirements.txt

6. Sentiment Analysis Tool Evaluation

Numerous sentiment analysis tools are available, including many specifically designed for the software engineering (SE) domain. With such a variety of tools, it can be challenging to choose a good fit.

In this chapter, I will describe some of the popular sentiment analysis tools, highlight their strengths and weaknesses, and then test some promising ones to try and find out if there are suitable tools for performing sentiment analysis on developer community content (See RQ2 in 1.1).

Then I will explore if there is a sentiment trend in a subset of my data to see if developers emotions towards MSA might have changed over the years (See RQ3 in 1.1).

6.1. Comparison of Sentiment Analysis Tools in Literature

As discussed in Section 3.2, general off the shelf sentiment analysis tools perform worse then tools specifically designed for the SE domain.

However, [LZB$^+$18] points out that while retraining sentiment analysis tools for SE tasks is necessary, it does not automatically guarantee reliability in all use cases.

For instance, these tools can be domain sensitive. This means that a tool trained with Jira data may not work well for Stack Overflow data without additional training.[LZB$^+$18]

This tells us that scores from different studies cannot be directly compared, as variations in data and setups might influence accuracy. However, metrics like F1 can still provide a general sense of whether a tool might perform well for our purposes.

I researched sentiment tools in the literature to find the most suitable tools for my study. A suitable tool should demonstrate respectable accuracy in studies, be straightforward to setup locally, and require minimal computing resources to ensure my local setup doesn't crash.

Here is an overview of some of the most popular sentiment analysis tools.

SentiStrength-SE

SentiStrength-SE is a lexicon-based sentiment analysis tool based on the widely used general SentiStrength tool. It refines SentiStrength by adjusting SE-domain-specific terms, resulting in improved performance on SE content. The F1 score improved from 62 % for Sentistrength to 77% for SentiStrengthSE. [IZ17]

The setup of SentiStrength-SE is straightforward, as it is available as a.jar file that can be executed directly over a GUI [IZ17].

SentiCR

SentiCR is a supervised sentiment analysis tool, which is specialized for the SE domain as it was trained on code review comments. The highest accuracy reached SentiCR on negative review comments with 83%[ABIR17].

To set up SentiCR on my local machine I had to make many changes to the code to fix deprecated libraries. I am unsure whether these changes might affect the accuracy of the tool or lead to problems in the execution, so I'm deciding against using SentiCR.

Senti4SD

Senti4SD is a sentiment analysis tool that was trained on data from Stack Overflow to capture the specific language used in the Software Engineering domain.

It combines lexicon-based, keyword-based, and word-embedding features to improve sentiment detection.

Lexicon-based features use predefined word lists with assigned sentiment scores to determine if a word carries positive, negative, or neutral sentiment. To figure out a sentence's overall score, the positive, negative, and neutral words are counted, and the most common type determines the sentiment.

Keyword-based features detect specific terms or expressions that are commonly used in the domain. For instance, if we have the sentence "I will debug this later" a lexikon-based tool might consider this as negative, because "debug" is associated with issues and might be assigned a negative sentiment in the dictionary. A keyword-based tool can look at the whole sentence and see that it isn't used negatively in this case, but rather describes a task.

Word embeddings analyze terms and word patterns to understand how different words combine to form meaning in context. For example "When the server crashed, I quickly implemented a fix, and everything is running smoothly now" would be interpreted as positive with word embeddings (due to the successful resolution), but as neutral or slightly negative with lexicon-based and keyword-based methods (focusing on "crashed" and "fix").

Using the combination of these 3 methods the researchers managed to reach a overall F1 of 0.87 with Senti4SD, which makes Senti4SD a promising candidate for my study.

[CLMN18]

I successfully set up Senti4SD after making minor adjustments, such as increasing the heap space and converting files to Unix format.

Bert4SentiSE

Bert4SentiSE is an adapted BERT model that has been specialized for the software development (SE) domain through fine-tuning. It performed well on a StackOverflow dataset with a F-measure of 0.87. [BKPVS20]

I faced some challenges while trying to set up Bert4SentiSE locally, like having to install extra programs not mentioned in the README. Once I got it running, the script had warnings about outdated libraries, and the training was too heavy for my computer, so I couldn't test BERT4SentiSE properly.

Because of the setup issues, I decided not to use BERT4SentiSE.

LLM

Large Language Models (LLM) could be used for SA. Smaller large language models (sLLMs) have potential in software engineering tasks, but they often run into issues because they lack enough labeled data. In comparison, big large language models (bLLMs) tend to perform well on datasets with limited training data and uneven distributions, even managing to do a great job in zero-shot situations. However, when there's plenty of training data or the dataset is more balanced, fine-tuned sLLMs can outperform bLLMs. [ZITL23]

The setup and running of the LLMs of [ZITL23] was too resource-intensive for my system, so I didn't evaluate them further. However, as ChatGPT is a powerful and easy-to-use LLM, I looked into that as a sentiment tool.

ChatGPT

ChatGPT is a widely used LLM. The easy-to-use web interface and flexible prompting makes ChatGPT a suitable candidate for testing out sentiment analysis.

The AI Model is based on based on the Transformer architecture, which means that it uses a neural network and language patterns. ChatGPT is trained on publicly available content from the internet, but also licensed third-party sources.

In the paper [MLAA24] ChatGPT with GPT 4 was used successfully for sentiment analysis and had good overall results (F1 of 0.85), it performed even slightly better than a fine-tuned BERT model, which had a F1 0f 0.83. It is relevant to note though that this study was not done on SE specific content and therefore results in the SE domain might be lower.[MLAA24]

Hybrids

While hybrid models that combine multiple tools can sometimes enhance performance, a simple majority voting system doesn't deliver optimal results according to [UGKR22]. SentiSead-Roberta was created as a hybrid model and achieved an F-score of 0.805, which is quite strong. However, in the same study, the standalone RoBERTa model reached an F-score of 0.801.[UGKR22] Given the only minor performance difference and RoBERTa's easier setup, it's more practical to use RoBERTa alone for nearly the same effectiveness with much less complexity.

Roberta

RoBERTa is a pre-trained Transformer model known for its advanced natural language processing capabilities and improvements over BERT. When fine-tuned for sentiment analysis in Software Engineering (SE), RoBERTa outperformed both traditional SE sentiment analysis tools and BERT across multiple datasets, achieving the best results in four out of six datasets. RoBERTa had its lowest F1 with 0.61 on app reviews, while its highscore was 0.92 on Github texts. This shows the domain sensitivity.[ZXT+20]

Setting up RoBERTa-Base is straightforward, but fine-tuning requires developing scripts for large labeled datasets [ZXT+20]. Unfortunately, [ZXT+20] only released their scripts, not the already fine-tuned model in the Hugging Face library. I attempted to run their scripts directly, but encountered issues.

Luckily, I then discovered EASTER, which contained a already finetuned RoBERTa.[SSG+22]

EASTER

The EASTER framework combines RoBERTa, which understands the context of words in sentences, with TextCNN, a convolutional neural network that uses deep learning for sentiment classification and identifies key phrases that influence sentiment.

The researchers fine-tuned the cardiffnlp/twitter-roberta-base-sentiment model on a Stack Overflow dataset, so RoBERTA can better understand word relationships in SE-specific user-generated content.

By combining RoBERTa with TextCNN, EASTER was able to reach impressive results, outperforming models like Senti4SD and SentiStrength-SE, with an overall accuracy of 97.04% in a StackOverflow dataset. EASTER also showed better generalizability across various datasets. [SSG+22]

The local setup of the tool worked quite well, it was well-documented and easy to use.

6.1.1. Tool Decision

Among the most interesting tools I evaluated are Senti4SD, EASTER, and ChatGPT, as these tools were easy enough to set up and demonstrated good results in the literature.

Senti4SD reached accuracy of up to 87% in a study done by its developers. [CLMN18].

As Sentistrength-SE's results in its original study were only 77% [IZ17] and in a study by the EASTER reseachers it was outperformed by both Senti4SD and EASTER [SSG+22], I decided against Sentistrength-SE.

EASTER has outperformed both Senti4SD and SentiStrength-SE and reached F1-scores of over 0.9 on some datasets, according to the findings in [SSG+22]. Since EASTER also utilizes RoBERTa and demonstrated strong results in that study, it appears to be the most promising option.

ChatGPT is interesting because the webinterface makes it easy to use. Also, I think that LLMs could be good in identifying if a sentiment is really connected to MSA or to another topic, as I can specify that with a prompt.

I want to remind that tools can perform differently in different domains.[LZB+18] None of the studies mentioned tested the tools on Reddit or Hackernews data. Therefore, before selecting a tool for a specific use case, it is important to first test different options on a small dataset of domain-specific data.[NGL18]

That's why I want to try out Senti4SD, EASTER and ChatGPT on a part of my dataset and evaluate their accuracy to find out if they might be suitable to the the sentiment analysis on my dataset (See RQ2 in 1.1).

6.2. Tool Accuracy of Sentiment Analysis Tools on my dataset

To find out which tool gives the most accurate sentiment predictions on my dataset, I want to compare the predictions of Senti4SD, EASTER and ChatGPT to my own manual predictions.

I created a test dataset [1] that consists of some comments from two randomly chosen discussions per source, so 4 discussions in total. I manually classified around 50 comments per data source, so around 100 comments in total.

To evaluate the tool accuracy I use the metrics Precision, Recall and F1 (See 3.3). To calculate these metrics automatically, I created Python scripts[2]. These scripts use the ‘sklearn. metrics‘ library[3] to compute the metrics, which I then save to a file and also print to the console.

6.2.1. Manual Classification

I manually classified the 100 comments rating the sentiment from -1 (negative), 0 (neutral) to 1(positive). I only considered the feeling towards MSA, not emotions for other things like monoliths.

I noticed that classifying the comments was quite challenging. Many comments are complex and long, making it hard to identify a clear overall emotion. Additionally, some comments may express positive emotions, but not towards MSA, but rather towards unrelated concepts like other architectures or their workplace.

The result of this manual classification is a dataset [4] of around 100 comments with sentiment labels (-1,0,1).

6.2.2. How did I use the tools?

EASTER Usage

The researchers behind EASTER[SSG+22] have published their implementation as a reproducible package[5]. I cloned their repository and installed the required dependencies. I created a virtual environment with Python 3.9 for this, as this is the recommended version by EASTER.

Using EASTER’s ‘predict()‘ function in the EASTER.py [6], I predicted sentiments on

[1]https://github.com/andreasw21/microserviceAnalysis/blob/master/Tool_Accuracy_Analysis/Labelled_Discussions/Manually/initial_labelled_comments.csv

[2]https://github.com/andreasw21/microserviceAnalysis/tree/master/Tool_Accuracy_Analysis/Analyze_Sentiment_Results

[3]https://scikit-learn.org/1.5/modules/model_evaluation.html

[4]https://github.com/andreasw21/microserviceAnalysis/blob/master/Tool_Accuracy_Analysis/Labelled_Discussions/Manually/initial_labelled_comments.csv

[5]https://github.com/xiaobo-lab/EASTER

[6]https://github.com/xiaobo-lab/EASTER/blob/main/EASTER.py

my dataset and then calculated[7] the accuracy of the predictions compared to my manual classifications.

ChatGPT Usage

I used ChatGPT 4o over the web interface [8] as this is the easiest way. According to the study by [MLAA24], a one-shot approach is recommended for optimal performance with ChatGPT in sentiment analysis tasks. A one-shot approach means that the model is instructed on how to solve a specific task using one example, that shows how an example sentence should be classified. I used a structured prompt to analyze sentiment specifically related to "Microservice Architecture", which is based on the prompt used in [MLAA24]:
" **Persona:** You are an NLP analyst tasked with understanding public sentiment on Microservice Architecture as an architectural style.

Instruction: Analyze the sentiment in the provided posts towards "Microservice Architecture" specifically. Use a sentiment scale of -1 (negative), 0 (neutral), and 1 (positive). Only consider opinions directly related to Microservice Architecture, ignoring any unrelated sentiments.

Output: The output should include: 1. The original CSV file with an additional column indicating the determined sentiment. 2. A summary with sentiment counts listed vertically. "
Example: Post: "I love monoliths, they are the best. Monoliths improve everything, I love them! I also think microservices introduce unnecessary complexity to software projects." Sentiment: -1 (negative) – The sentiment reflects a negative view of microservices, specifically citing unnecessary complexity. Positive sentiments about monoliths are irrelevant and should be ignored.

Constraints: - Only analyze sentiments specifically regarding Microservice Architecture. - Exclude opinions that do not directly reference microservices. - Use the sentiment scale: -1 (negative), 0 (neutral), 1 (positive).

Senti4SD Usage

I downloaded Senti4SD from its GitHub repository[9] and used the Windows Subsystem for Linux (WSL)[10] for all Senti4SD-related tasks, because the recommend setup steps contained Linux commands. After installing all required dependencies in WSL, I ran the ‘classification.sh‘ script on my dataset.

The script produced an unordered list of results, which I needed to process further. To organize the output, I wrote a Python script[11] that sorts the results and saves them as a

[7] `https://github.com/andreasw21/microserviceAnalysis/blob/master/Tool_Accuracy_Analysis/Analyze_Sentiment_Results/generate_analysis.py`
[8] https://chatgpt.com/?model=gpt-4o (Accessed 7.11.2024)
[9] `https://github.com/collab-uniba/Senti4SD/tree/master`
[10] `https://learn.microsoft.com/en-us/windows/wsl/install`
[11] `https://github.com/andreasw21/microserviceAnalysis/blob/master/Tool_Accuracy_Analysis/Postprocess_Senti4SD/senti4SD_postprocessing.py`

CSV. The final file contains the original input data and adds Senti4SD classifications as a new column.

6.3. Results

I compared Senti4SD, EASTER and ChatGPT to my manual predictions and calculated [12] Precision, Recall and F1 for them.

6.3.1. Senti4SD Results

Senti4SD reached an F1 score of 0.45, as you can see in 6.1. This score is not high enough to rely on for accurate predictions. The F1 score is fairly consistent across sentiment classes, with values of 0.39 for positive, 0.49 for negative, and 0.46 for neutral.

Sentiment Class	**Precision (P)**	**Recall (R)**	**F1-Score (F)**
Negative (-1)	0.56	0.43	0.49
Neutral (0)	0.36	0.64	0.46
Positive (1)	0.48	0.33	0.39
Macro Average	0.47	0.47	0.45
Weighted Average	0.48	0.45	0.45

Table 6.1.: Senti4SD Performance Metrics by Sentiment Class

6.3.2. EASTER Results

Sentiment Class	**Precision (P)**	**Recall (R)**	**F1-Score (F)**
Negative (-1)	0.89	0.60	0.71
Neutral (0)	0.38	1.00	0.55
Positive (1)	1.00	0.10	0.18
Macro Average	0.76	0.57	0.48
Weighted Average	0.79	0.55	0.51

Table 6.2.: EASTER Performance Metrics by Sentiment Class

The results in Table 6.2 show that EASTER performs well in detecting negative sentiments. For positive sentiments, the results are mixed: the precision is high (1.00), but the recall is low (0.10). My interpretation of these results is that while the positive instances EASTER identifies are indeed positive, it fails to capture most positive cases.

In contrast, the high recall for neutral sentiments (1.00) suggests that EASTER successfully identifies nearly all neutral instances. However, its low precision in this

[12] `https://github.com/andreasw21/microserviceAnalysis/blob/master/Tool_Accuracy_Analysis/Analyze_Sentiment_Results/generate_analysis.py`

category (0.38) indicates a high rate of false positives, meaning that EASTER incorrectly classifies many cases as neutral.

Overall, the weighted average F1-Score of 0.51 indicates limited performance, but is higher than the one from Senti4SD, which was 0.45.

6.3.3. ChatGPT Results

The accuracy metrics achieved by ChatGPT compared to my manual classification is shown in Table 6.3

Sentiment Class	Precision (P)	Recall (R)	F1-Score (F)
Negative (-1)	0.70	0.17	0.27
Neutral (0)	0.33	0.56	0.41
Positive (1)	0.39	0.57	0.46
Macro Average	0.47	0.43	0.38
Weighted Average	0.51	0.39	0.36

Table 6.3.: ChatGPT Performance Metrics by Sentiment Class

These results indicate that ChatGPT's performance with this setup and prompts is low as the weighted average F1 is 0.36. It performed best on positive posts (F1 0.46) and worst on negative (0.27)

The overall weighted F1-Score of 0.36 suggests that the model struggles with the sentiment analysis on our dataset even more than EASTER and Senti4SD.

6.3.4. Result Discussion

When comparing the accuracy of the tested tools, it became visible that EASTER performed best with a weighted F1-Score of 0.51, while Senti4SD achieved an F1-Score of 0.45, and ChatGPT scored 0.36 as seen in 6.4

All of these results are worse than the results reported for these tools in the literature. Both Senti4SD and EASTER had F1 scores of around 0.9 on a StackOverflow dataset in a prior study [SSG$^+$22].

However, the researchers in the study [SSG$^+$22] trained their tools on StackOverflow data, which was similar to the data on which they evaluated the tools. My dataset on the other hand is quite different from their training set.

My texts are longer than those used in [SSG$^+$22]. The increased length also brings more complexity, as the discussions about MSA are often full of technical details.

An additional challenge is that I am only looking for sentiments related to MSA. The texts often contain sentiments about other topics, which can mislead the tools.

The results in [SSG$^+$22] showed that tool accuracy can vary significantly when data from different domains is used. For example, on app reviews, Senti4SD achieved an accuracy of almost 64%, while EASTER reached 75%, both significantly below the 0.9 scores reported on the StackOverflow dataset. On JIRA issues, Senti4SD performed even worse, with an accuracy of 58%. [SSG$^+$22]

I did not find any accuracy evaluations for EASTER and Senti4SD on software engineering-specific Reddit or HackerNews posts. Therefore, my relatively low results could also be attributed to domain specificity.

In [SSG$^+$22], they found that EASTER did better on bigger datasets. With just 1,500 comments, the accuracy was around 0.8, but when they used a larger dataset of 4,423 comments, accuracy reached up to 0.97. s

In my case, I'm working with only 100 comments, which is much less than the 4423 used in their studies [SSG$^+$22] and this could have a significant impact on my results.

Furthermore, I noticed that tool accuracy varied across different sentiment classes. In my results, EASTER achieved an F1-Score of 0.71 on negative posts but only 0.18 on positive posts as seen in 6.2. While this variation is surprising, such differences between sentiment classes are consistent with the results from [SSG$^+$22]. For instance, on the StackOverflow 1500 dataset, EASTER achieved an F1-Score of 0.88 on neutral posts but only 0.4 on negative posts [SSG$^+$22].

Model	**F1-Score (Macro)**	**F1-Score (Weighted)**
Senti4SD	0.45	0.45
EASTER	0.48	0.51
ChatGPT	0.38	0.36

Table 6.4.: Performance Metrics for Senti4SD, EASTER, and ChatGPT

6.4. Improvement

So far the most accurate sentiment tool in my study was EASTER, with a F1 of 0.51. I want to try out some changes to my dataset and see if this can improve accuracy.

Improvement Method	**Senti4SD**	**EASTER**	**ChatGPT**
No Improvement (Baseline)	0.45	**0.51**	0.36
Shorter than 50 words	0.33	**0.63**	0.36
Only sentences with "Microservices"	0.26	0.51	0.41
ChatGPT Compression	0.34	0.25	0.38
Combined Approach (All Tools)	**0.66**	**0.66**	**0.66**

Table 6.5.: Weighted F1 Scores of Models with Various Improvement Methods

Comments shorter than 50 words

I suspect that the tools struggle with the long length of my comments. That's why I created a new dataset, where I removed all comments that have more than 50 words.[13] I

[13]`https://github.com/andreasw21/microserviceAnalysis/blob/master/Tool_Accuracy_Analysis/Improvement_Ideas/less_than_x_words.py`

tested out different word limits (25, 50, 100, 200), but I had the best results with a limit of 50 words per comment. This left 45 comments in my manually labeled dataset.

I tested the accuracy of the three tools again. Senti4SD performed worse, with its weighted F1-Score dropping from 0.45 to 0.33. ChatGPT remained consistent, maintaining a F1-Score of 0.36. EASTER showed improvement and achieved a weighted F1-Score of 0.63, which is the highest score I have reached so far.

Then I stacked up the manually classified dataset up to 100 comments again, but again only comments with less than 50 words. I tested EASTER's accuracy again and reached a F1 of 0.61, which is consistent with the 0.63 I reached on the 45 comments.

Only sentences with "Microservices"

Another reason I think my results were poor is that, in many cases, there was sentiment present, but it was not directly related to MSA. To increase the likelihood of capturing MSA-related sentiments in a fast and easy way, I post-processed my dataset so that only sentences containing the term "microservice*" were retained in the comments[14] . This adjustment improved ChatGPT's F1-Score from 0.36 to 0.41, while EASTER remained unchanged from the baseline (0.51). However, Senti4SD's performance worsened, dropping from 0.45 to 0.26. These results suggest that this method might remove too much context so that the tools actually don't have it easier in finding overall sentiments.

ChatGPT Compression

I asked ChatGPT to compress my comments by summarizing the parts with sentiment towards MSA and leaving out anything irrelevant. The prompt I used for this can be found in [15] and can be summed up like this:

- Shorten the comments while preserving the emotions.
- Use simpler and more concise sentence structures.
- Exclude parts of the comment that are unrelated to microservices. For example, get rid of paragraphs about monoliths or people.
- Ensure that each comment contains a maximum of 30 words.

The problem is that ChatGPT tends to write everything in a pretty neutral tone even when repeatedly prompting it to keep emotions intact, which makes it harder for the tools to detect sentiments in the resulting texts. As a result Senti4SD and EASTER performed worse than the baseline, while ChatGPT only improved slightly, going from an F1-Score of 0.36 to 0.38.

[14] `https://github.com/andreasw21/microserviceAnalysis/blob/master/Tool_Accuracy_Analysis/Improvement_Ideas/remove_irrelevant_sentences.py`

[15] `https://github.com/andreasw21/microserviceAnalysis/blob/master/Tool_Accuracy_Analysis/Improvement_Ideas/chatGPT_summup_prompts.txt`

Combined approach

I wanted to try out an approach where I could combine the strengths of EASTER, ChatGPT, and Senti4SD into a unified sentiment score. I put all the predictions into one file and wrote a Python script to calculate a combined sentiment [16] (See 6.1)

```
if easter_sentiment == -1:
   combined_sentiment = -1 # Trust EASTER if it says -1

elif (easter_sentiment == 0 and senti4sd_sentiment == -1 and
    chatgpt_sentiment == -1):
   combined_sentiment = -1

elif ((easter_sentiment in (0, 1)) and (senti4sd_sentiment in (0, 1)) and
     chatgpt_sentiment == 1):
  combined_sentiment = 1 # Trust ChatGPT if it says 1

else:
    combined_sentiment = easter_sentiment # Otherwise, trust EASTER
```

Listing 6.1: Code

For this I used the "Shorter than 50"-dataset that I filled up to 100 manually classified comments.

Since EASTER had achieved the best results on the "shorter than 50" dataset, I decided to trust its predictions most.

However, EASTER performed poorly on positive sentiment, so I chose to rely on ChatGPT for the positive predictions most. For ChatGPT, I used the "Only sentences with 'Microservices'" method on my dataset to improve it's accuracy.

As for Senti4SD, I used its predictions to weigh in on combined sentiment decisions to help weigh out disagreements between the tools.

This combined approach achieved a new F1-score high of 0.66.

6.4.1. Discussion of improved results

The results in Table 6.5 show that the best improvement with a single tool was the "Shorter than 50 words" method, achieving a weighted F1 score of 0.63 with EASTER. This suggests that comment length was a key challenge for EASTER, as shorter comments seemed to work better for it.

The best overall result came from the combined approach, which shows great potential. It could maybe be even better if different combination methods are tested. I tried a few different combinations, and this was the best so far.

The combined approach gave us an F1 score of 0.66. This means it can be useful for drawing some general conclusions from the dataset, but we can't fully trust it for very specific predictions just yet.

[16] https://github.com/andreasw21/microserviceAnalysis/blob/master/Tool_Accuracy_Analysis/Labelled_Discussions/All_Tools_Combined/combined_sentiment.py

In answer to RQ2 (1.1) it can be said that EASTER is the most suitable single tool of the tools I considered. While EASTER's results as a single tool without any improvements reached an F1 of only 0.51, it did perform reasonably well for detecting negative sentiments, with an F1 score of 0.71 for negative. This means that EASTER could at least be used to identify the proportion of negative comments. However, when it comes to determining whether something is positive or neutral, we can't rely on EASTER as a single tool enough. With the "Shorter than 50" improvement EASTER reached a F1 of 0.63, which made it the most accurate of my tested single tools.

6.5. Sentiment Trend Exploration

This section discusses if according to a subset of my data there is any indication that suggests a change in developers' sentiments expressed in online communities over the years. 1.1

I analyzed a part of the dataset to explore sentiment trends over different years. To do this, I selected the oldest discussions from my dataset, which were from 2015, and the newest, from 2024.

I wanted to use a single tool to minimize effort, as combining tools would require more work for only a slight improvement in accuracy (0.63 vs. 0.66). Therefore I used EASTER for the predictions, as it had the highest accuracy of the single tools. I chose to only consider comments shorter than 50 words, as this improved EASTER's accuracy.

Even though EASTER's accuracy isn't perfect (0.63), I hope that when the same tool is used to predict 2 different datasets it might make similar prediction mistakes so some mistakes might also cancel each other out.

I found 161 comments from 2024 that were shorter than 50 words, but only 29 from 2015. Therefore, I decided to include comments from 2016 and 2017 with the 2015 comments.

I used EASTER to predict the sentiments in my two subsets: [17] [18].

Then, I counted the positive, negative, and neutral sentiments: [19] [20].

Figure 6.1 shows the accuracy results of the improvement methods.

The negative class increases in the newer data, from 16.67% in 2015-2017 to 25.47% in 2024. As a result, the neutral class decreased from 76.67% to 67.70%, while the positive class remained almost unchanged.

These results suggest that there might be a growing dislike for microservice architecture among developers over the years, answering RQ3 1.1 with a tentative "yes". This negative trend aligns with the trend in companies moving away from MSA recently as I discussed in 2.1

[17]https://github.com/andreasw21/microserviceAnalysis/blob/master/Tool_Accuracy_Analysis/Sentiment_Trend/2015-2017/2015-2017_less_than_50_words.csv

[18]https://github.com/andreasw21/microserviceAnalysis/blob/master/Tool_Accuracy_Analysis/Sentiment_Trend/2024/2024_less_than_50_words.csv

[19]https://github.com/andreasw21/microserviceAnalysis/blob/master/Tool_Accuracy_Analysis/Sentiment_Trend/2015-2017/sentiment_analysis_summary.txt

[20]https://github.com/andreasw21/microserviceAnalysis/blob/master/Tool_Accuracy_Analysis/Sentiment_Trend/2024/sentiment_analysis_summary.txt

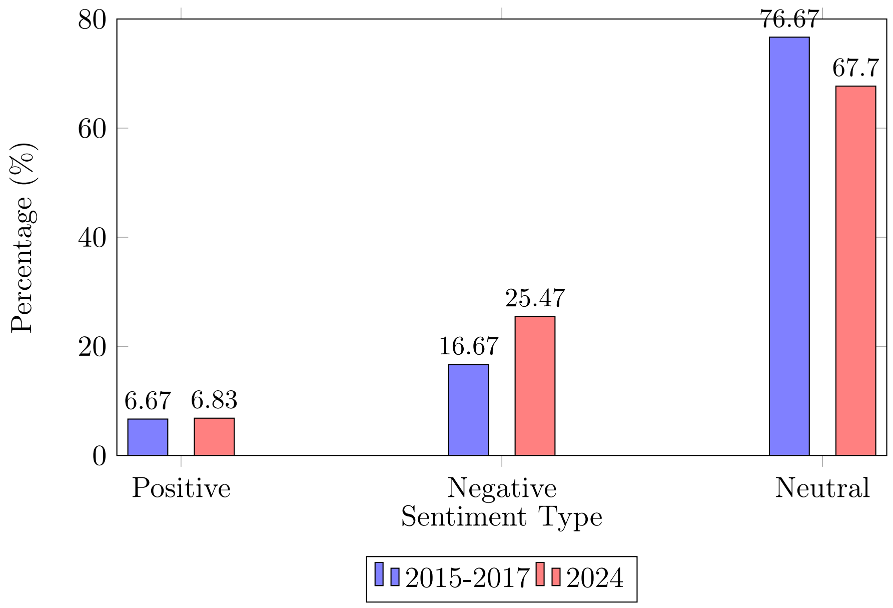

Figure 6.1.: Comparison of sentiment distribution between 2015-2017 and 2024.

However, since EASTER's accuracy is only 0.63 on data with a wordlimit of 50 words, the reliability of these results cannot be guaranteed. Additionally, this analysis is based on a subset of the data, so examining the full dataset could lead to different results.

Nonetheless, this trend makes future research into this negative shift interesting, as it suggests there may be a trend to find.

6.6. Future Work & Limitations

In future work it would be useful to further improve the accuracy of the sentiment analysis tools. Future improvements that could be explored include:

- **Improving ChatGPT prompts**
- **Using Named Entity Recognition (NER)** to identify MSA-related sentences
- **Further refining the combination of tools** to better use their individual strengths

It would also be interesting to evaluate a larger dataset. This would require manually labeling more data. The study in [SSG$^+$22] used StackOverflow datasets with 1,500 and 4,423 texts, while I only worked with 100 posts. It would be valuable to see how a larger dataset impacts accuracy.

Additionally, cross-evaluating my manual labeling could be helpful. Having multiple people label the data and compare their results would help ensure consistency. Since my manual labels are used as the ground truth, any mistakes in my labeling affect the results.

Once a sufficiently high accuracy is achieved, the goal would be to use the tool to predict sentiments across the entire dataset. This could solidify or disprove the sentiment trend identified in the subset of data (see Section 6.5), which suggested a negative shift.

By doing so, we could be able to find out if developers' feelings towards MSA have actually changed over the years.

7. Conclusion

This thesis explored the first steps in analyzing whether the trend of companies like Amazon moving away from Microservice Architecture (MSA) [SLT23] is also visible in online developer communities. The idea is to use sentiment analysis tools to classify comments in online discussions as negative, neutral, or positive.

I evaluated different online communities as data sources and chose Reddit and Hacker-News. These platforms were best suited for my use-case, because they have free, easy-to-use APIs and many discussions about MSA that also contain emotions. (RQ1, 1.1)

I built a tool to collect discussions from these platforms and saved each discussion and its comments in a CSV file. A discussion contains one thread of Reddit/Hacker-News, with the orignal-post and the comments, but also post-id, creation-data and datasource. I tried to make sure only discussions about MSA were included by searching for the word "microservice" and then applying filters to remove irrelevant results, like overly technical posts that focused on specific technologies instead of the general pros and cons of MSA. This dataset could then be used for sentiment analysis.

Before analyzing sentiment, I needed to find suitable tools. Many sentiment analysis were not developed for the software engineering (SE) domain and therefore can not provide accurate results on my data. I read different papers about sentiment tools in SE and chose to test three tools, that were possible for me to setup locally and showed good results in literature: Senti4SD, EASTER, and ChatGPT. (RQ2, 1.1)

EASTER performed the best among the three tools in an evaluation using 100 manually labeled comments, achieving a weighted F1 score of 0.51 compared to my manual classifications. This score is not high enough to make reliable predictions. The complexity of my dataset, which contains long comments written in technical language and comments with emotions about topics other than MSA, contributed to this low accuracy.

I tested different dataset changes to improve accuracy. The best result for a single tool came from only using comments under 50 words, raising accuracy to 0.63. Combining the strengths of all three tools brought the F1-score to 0.66.

Finally, I used EASTER to check if developer sentiment about MSA has changed over time (RQ3, 1.1). Using a subset of my data with comments shorter than 50 words to improve accuracy, I compared sentiment from 2024 to the period 2015-2017. The results showed more negative sentiments in 2024 (25%) compared to 2015-2017 (17%), with neutral sentiments dropping from 77% to 67%, while positive sentiments stayed at around 7%. These results suggest that developers indeed feel more negative towards MSA now than they did 2015-2017. However, since this analysis only used a subset of the data, and EASTER's accuracy is still limited, the results are not enough to confirm a clear trend.

Still, the findings suggest that future research into sentiment trends about MSA using online developer communities could be interesting.

Bibliography

[ABIR17] Toufique Ahmed, Amiangshu Bosu, Anindya Iqbal, and Shahram Rahimi. SentiCR: A Customized Sentiment Analysis Tool for Code Review Interactions. In *32nd IEEE/ACM International Conference on Automated Software Engineering (NIER track)*, ASE '17, 2017.

[AMPJ17] Carlos M. Aderaldo, Nabor C. Mendonça, Claus Pahl, and Pooyan Jamshidi. Benchmark requirements for microservices architecture research. In *Proceedings of the 1st International Workshop on Establishing the Community-Wide Infrastructure for Architecture-Based Software Engineering*, ECASE '17, page 8–13. IEEE Press, 2017.

[BJMH15] Titus Barik, Brittany Johnson, and Emerson Murphy-Hill. I heart hacker news: expanding qualitative research findings by analyzing social news websites. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, ESEC/FSE 2015, page 882–885, New York, NY, USA, 2015. Association for Computing Machinery.

[BKPVS20] Eeshita Biswas, Mehmet Efruz Karabulut, Lori Pollock, and K. Vijay-Shanker. Achieving reliable sentiment analysis in the software engineering domain using bert. In *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 162–173, 2020.

[BMPM19] Alan Bandeira, Carlos Alberto Medeiros, Matheus Paixao, and Paulo Henrique Maia. We need to talk about microservices: an analysis from the discussions on stackoverflow. In *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*, pages 255–259, 2019.

[CLMN18] Fabio Calefato, Filippo Lanubile, Federico Maiorano, and Nicole Novielli. Sentiment polarity detection for software development. *Empirical Software Engineering*, 23(3):1352–1382, 2018.

[GH23] Shiramshetty Gouthami and Nagaratna Hegde. *Automatic Sentiment Analysis Scalability Prediction for Information Extraction Using SentiStrength Algorithm*, pages 21–30. 03 2023.

[GKG22] Dimitrios Gravanis, George Kakarontzas, and Vassilis Gerogiannis. You don't need a microservices architecture (yet): Monoliths may do the trick. In *Proceedings of the 2021 European Symposium on Software Engineering*, ESSE '21, page 39–44, New York, NY, USA, 2022. Association for Computing Machinery.

[Ham23] Muhammad Hamza. Transforming monolithic systems to a microservices architecture. *SIGSOFT Softw. Eng. Notes*, 48(1):67–69, jan 2023.

[IZ17] Md Rakibul Islam and Minhaz F. Zibran. Leveraging automated sentiment analysis in software engineering. In *Proceedings of the 14th International Conference on Mining Software Repositories*, MSR '17, page 203–214. IEEE Press, 2017.

[JDS15] Robbert Jongeling, Subhajit Datta, and Alexander Serebrenik. Choosing your weapons: On sentiment analysis tools for software engineering research. In *2015 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 531–535, 2015.

[KSNNSN24] Joao Paulo Karol Santos Nunes, Shiva Nejati, Mehrdad Sabetzadeh, and Elisa Yumi Nakagawa. Self-adaptive, requirements-driven autoscaling of microservices. In *Proceedings of the 19th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS '24, page 168–174, New York, NY, USA, 2024. Association for Computing Machinery.

[LZB$^+$18] Bin Lin, Fiorella Zampetti, Gabriele Bavota, Massimiliano Di Penta, Michele Lanza, and Rocco Oliveto. Sentiment analysis for software engineering: how far can we go? In *Proceedings of the 40th International Conference on Software Engineering*, ICSE '18, page 94–104, New York, NY, USA, 2018. Association for Computing Machinery.

[MLAA24] Lany Laguna Maceda, Jennifer Laraya Llovido, Miles Biago Artiaga, and Mideth Balawiswis Abisado. Classifying sentiments on social media texts: A gpt-4 preliminary study. In *Proceedings of the 2023 7th International Conference on Natural Language Processing and Information Retrieval*, NLPIR '23, page 19–24, New York, NY, USA, 2024. Association for Computing Machinery.

[NGL18] Nicole Novielli, Daniela Girardi, and Filippo Lanubile. A benchmark study on sentiment analysis for software engineering research. In *Proceedings of the 15th International Conference on Mining Software Repositories*, MSR '18, page 364–375, New York, NY, USA, 2018. Association for Computing Machinery.

[PZZ$^+$22] Xin Peng, Chenxi Zhang, Zhongyuan Zhao, Akasaka Isami, Xiaofeng Guo, and Yunna Cui. Trace analysis based microservice architecture measurement. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ESEC/FSE 2022, page 1589–1599, New York, NY, USA, 2022. Association for Computing Machinery.

[RWW$^+$18] Zhongshan Ren, Wei Wang, Guoquan Wu, Chushu Gao, Wei Chen, Jun Wei, and Tao Huang. Migrating web applications from monolithic structure to microservices architecture. In *Proceedings of the 10th Asia-Pacific Symposium on Internetware*, Internetware '18, New York, NY, USA, 2018. Association for Computing Machinery.

[SA24] Niklas Sänger and Sebastian Abeck. Fine-grained authorization in microservice architecture: A decentralized approach. In *Proceedings of the 39th ACM/SIGAPP Symposium on Applied Computing*, SAC '24, page 1219–1222, New York, NY, USA, 2024. Association for Computing Machinery.

[SC24] Arghavan Sanei and Jinghui Cheng. Characterizing usability issue discussions in open source software projects. *Proc. ACM Hum.-Comput. Interact.*, 8(CSCW1), April 2024.

[Sha21] Reshma Shaji. Exploratory data analysis on reddit data: An efficient pipeline for classification of flairs. In *2021 IEEE Seventh International Conference on Multimedia Big Data (BigMM)*, pages 65–68, 2021.

[SL24] Ruoyu Su and Xiaozhou Li. Modular monolith: Is this the trend in software architecture? In *Proceedings of the 1st International Workshop on New Trends in Software Architecture*, SATrends '24, page 10–13, New York, NY, USA, 2024. Association for Computing Machinery.

[SLT23] Ruoyu Su, Xiaozhou Li, and Davide Taibi. Back to the future: From microservice to monolith. In *2023 International Conference on Microservices*, Oulu, Finland, August 2023. University of Oulu and Tampere University.

[SSG$^+$22] Kexin Sun, Xiaobo Shi, Hui Gao, Hongyu Kuang, Xiaoxing Ma, Guoping Rong, Dong Shao, Zheng Zhao, and He Zhang. Incorporating pre-trained transformer models into textcnn for sentiment analysis on software engineering texts. In *Proceedings of the 13th Asia-Pacific Symposium on Internetware*, Internetware '22, page 127–136, New York, NY, USA, 2022. Association for Computing Machinery.

[Sto15] Greg Stoddard. Popularity and quality in social news aggregators: A study of reddit and hacker news. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15 Companion, page 815–818, New York, NY, USA, 2015. Association for Computing Machinery.

[UGKR22] Gias Uddin, Yann-Gaël Guéhénuc, Foutse Khomh, and Chanchal K. Roy. An empirical study of the effectiveness of an ensemble of stand-alone sentiment detection tools for software engineering datasets. *ACM Trans. Softw. Eng. Methodol.*, 31(3), April 2022.

[Wan19] Yuwei Wang. Towards service discovery and autonomic version management in self-healing microservices architecture. In *Proceedings of the 13th European Conference on Software Architecture - Volume 2*, ECSA '19, page 63–66, New York, NY, USA, 2019. Association for Computing Machinery.

[ZITL23] T. Zhang, I. Clairine Irsan, F. Thung, and D. Lo. Revisiting sentiment analysis for software engineering in the era of large language models. *arXiv preprint arXiv:2310.11113*, 2023.

[ZXT+20] Ting Zhang, Bowen Xu, Ferdian Thung, Stefanus Agus Haryono, David Lo, and Lingxiao Jiang. Sentiment analysis for software engineering: How far can pre-trained transformer models go? In *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 70–80, 2020.

A. Appendix

Code and presentation slides: https://github.com/andreasw21/microserviceAnalysis/tree/master