Bachelor's Thesis

Automated Topic Extraction in Software Engineering: A Large Language Model Approach to Analyzing Informal Practitioner Discourse

José Javier González

September 7, 2026

Advisors:

Marvin Wyrich	Chair of Software Engineering
Tobias Dick	Chair of Software Engineering

Reviewers:

Prof. Dr. Sven Apel	Chair of Software Engineering
Prof. Dr. Andreas Zeller	CISPA Helmholtz Center for Information Security





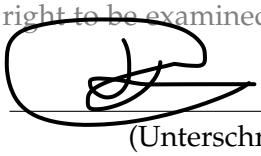
Chair of Software Engineering ◇ Saarland Informatics Campus ◇ Saarland University

José Javier González: *Automated Topic Extraction in Software Engineering: A Large Language Model Approach to Analyzing Informal Practitioner Discourse,* © September 2026

Erklärung Statement

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und ohne die Beteiligung dritter Personen verfasst habe, und dass ich keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Alle Stellen der Arbeit, die wörtlich oder sinngemäß aus Veröffentlichungen oder aus anderweitigen fremden Äußerungen entnommen wurden, sind als solche kenntlich gemacht. Insbesondere bestätige ich hiermit, dass ich bei der Erstellung der nachfolgenden Arbeit mittels künstlicher Intelligenz betriebene Software (z. B. ChatGPT) ausschließlich zur Textüberarbeitung/-korrektur und zur Code-Vervollständigung und nicht zur Bearbeitung der in der Arbeit aufgeworfenen Fragestellungen zu Hilfe genommen habe. Alle mittels künstlicher Intelligenz betriebenen Software (z. B. ChatGPT) generierten und/oder bearbeiteten Teile der Arbeit wurden kenntlich gemacht und als Hilfsmittel angegeben. Ich erkläre mich damit einverstanden, dass die Arbeit mittels eines Plagiatsprogrammes auf die Nutzung einer solchen Software überprüft wird. Mir ist bewusst, dass der Verstoß gegen diese Versicherung zum Nichtbestehen der Prüfung bis hin zum Verlust des Prüfungsanspruchs führen kann.

I hereby declare that I have written this thesis independently and without the involvement of third parties, and that I have used no sources or aids other than those indicated. All passages taken directly or indirectly from publications or other external sources have been identified as such. In particular, I confirm that I have used AI-based software (e.g., ChatGPT) exclusively for the following permitted sub-tasks: text rewriting/revision and code completion, and not to address or formulate the main research questions of the thesis. All parts of the thesis that were generated and/or edited using AI-based software (e.g., ChatGPT) have been disclosed and documented in accordance with the documentation requirements. I agree that the thesis may be checked using plagiarism detection software, including checks for the use of such software. I am aware that any violation of this declaration may result in failing the examination and lead to losing the right to be examined.

Saarbrücken, 07.09.2026 

(Datum Date) (Unterschrift Signature)

Einverständniserklärung (optional) Declaration of Consent (optional)

Ich bin damit einverstanden, dass meine (bestandene) Arbeit in beiden Versionen in die Bibliothek der Informatik aufgenommen und damit veröffentlicht wird.

I agree to make both versions of my thesis (with a passing grade) accessible to the public by having them added to the library of the Computer Science Department.

Saarbrücken, 07.09.2026 

(Datum Date) (Unterschrift Signature)

Abstract

Software engineering research and industry practice do not always align, and closing that gap requires some way of observing what practitioners actually discuss, rather than relying on citation counts or conference programs. Informal platforms like YouTube and Stack Exchange capture this discourse directly, but at a volume and pace that manual analysis cannot keep up with, and existing automated approaches to topic modeling either demand a fixed number of topics chosen in advance or struggle to capture the meaning behind short, noisy, conversational text. This thesis designs, implements, and evaluates a pipeline that extracts software engineering topics from YouTube and Stack Exchange using a locally hosted large language model, groups semantically similar topics through sentence embeddings and density based clustering, and tracks how each resulting topic cluster develops across a multi year window using statistical trend tests built for short series with real zero counts. The pipeline extracts and clusters topics reliably across both platforms, confirmed by manual review of extraction accuracy and cluster coherence, and applying it to discourse collected between 2020 and 2026 reveals a topic landscape built around a stable core of long running themes, layered with clear directional movement, some topics clearly rising, most visibly toward AI assisted and agentic coding tools, others clearly fading, each classification carrying an explicit, honestly reported level of confidence. The central contribution of this thesis is not any single trend it reports, but the pipeline itself, a reusable instrument that can be pointed at a new collection of the same discourse at any future point to produce a comparably grounded read of how practitioner attention has moved on.

Acknowledgments

I would like to express a profound appreciation for both of my advisors Marvin Wyrich and Tobias Dick for their support and guidance in this work, without them none of this would have been possible.

I would also like to thank Prof. Dr. Sven Apel and Prof. Dr. Andreas Zeller for agreeing to review this work.

Finally, I would like to thank all friends and family members who supported me throughout this incredible journey.

Contents

1 Introduction 1
1.1 Goal of this Thesis . . . 2
1.2 Overview . . . 2
2 Background 3
3 Methodology 9
3.1 Research Questions . . . 9
3.2 Operationalization . . . 10
3.2.1 RQ1: Evaluating Pipeline Feasibility . . . 10
3.2.2 RQ2: Topic Frequency and Evolution . . . 11
3.3 Overview of the Pipeline . . . 12
3.4 Data Collection . . . 13
3.4.1 Stack Exchange Data . . . 13
3.4.2 YouTube Data . . . 13
3.5 Topic Extraction . . . 14
3.5.1 Model and Setup . . . 14
3.5.2 Data Sources and Large Language Model (LLM) Input Construction . . . 14
3.5.3 Prompt Design . . . 15
3.5.4 Iterative Prompt Refinement . . . 17
3.5.5 Output Format and Post-processing . . . 17
3.6 Embedding and Clustering of Topic Labels . . . 17
3.6.1 Normalization of Topic Labels . . . 18
3.6.2 Sentence Embeddings . . . 18
3.6.3 Dimensionality Reduction with Uniform Manifold Approximation and Projection (UMAP) . . . 18
3.6.4 Clustering with Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) . . . 18
3.6.5 From Topic Label Clusters to Item Clusters . . . 20
3.6.6 Naming Clusters . . . 25
3.6.7 Noise Handling . . . 25
3.7 Analyzing Topic Evolution . . . 25
3.7.1 Cluster Size and Yearly Share . . . 26
3.7.2 Topic Diversity (Entropy) . . . 27
3.7.3 Descriptive Statistics and Ranking of Clusters by Size . . . 27
3.7.4 Linear Regression . . . 28
3.7.5 Polynomial Regression (Degree 2) . . . 28
3.7.6 Mann-Kendall Trend Test . . . 28
3.7.7 Negative Binomial Growth Rate . . . 29

3.7.8 Secondary Growth Cross-Checks . . . 30
3.7.9 Combining the Trend Measures . . . 31
3.7.10 Descriptive Statistics and Ranking of Clusters by Trend Metrics . . . 32
3.7.11 Multiple Testing Correction . . . 32
4 Evaluation 33
4.1 Results . . . 33
4.1.1 Corpus Overview . . . 33
4.1.2 RQ1: Pipeline Feasibility . . . 34
4.1.3 RQ2: Topic Frequency and Evolution . . . 40
4.2 Discussion . . . 60
4.2.1 RQ1: Pipeline Feasibility . . . 60
4.2.2 RQ2: Topic Frequency and Evolution . . . 60
4.3 Threats to Validity . . . 63
4.3.1 Internal Validity . . . 63
4.3.2 External Validity . . . 64
5 Related Work 65
6 Concluding Remarks 69
6.1 Conclusion . . . 69
6.2 Future Work . . . 70
A Appendix 73
A.1 YouTube Channel List . . . 73
A.2 Polynomial Regression Fit Details . . . 75
A.3 Cross-Platform Cluster Overlap . . . 75
A.4 Trend Test P-Value Distributions . . . 77

Statement on the Usage of Generative Digital Assistants 79

Bibliography 81

List of Figures

Figure 3.1 Overview of the four pipeline stages . . . 12
Figure 3.2 A concrete example of the topic extraction step: an item's text is passed to the LLM, which returns a JSON list of extracted topic labels. 14
Figure 3.3 The three-step task decomposition used in both prompts, applied to a small worked example: candidate topics are noticed, duplicates are merged, and the least central remaining topic is dropped at selection. 16
Figure 3.4 An item's topic labels can land in different topic label clusters, producing multiple item cluster assignments for the same item. . . . 21
Figure 3.5 Item cluster membership is not exclusive: an item with several topics can belong to several item clusters at once. . . . 26
Figure 4.1 Distribution of topic label cluster sizes across all 2470 clusters . . . 38
Figure 4.2 Distribution of per cluster Density-Based Clustering Validation (DBCV) scores . . . 38
Figure 4.3 Distribution of mean membership probability per cluster . . . 39
Figure 4.4 Topic level cluster size, DBCV and membership probability shown together per cluster . . . 40
Figure 4.5 Distribution of item cluster size (total items) across all 2470 clusters 41
Figure 4.6 Cumulative share of item cluster assignments by cluster rank, largest to smallest . . . 44
Figure 4.7 Distribution of cluster lifespan, measured as number of active years out of seven . . . 44
Figure 4.8 Shannon entropy of the yearly item cluster assignment share, 2020 to 2026 . . . 45
Figure 4.9 Number of clusters per trend category, emerging, stable, declining and undetermined . . . 46
Figure 4.10 Trend category counts broken down by confidence tier, high, medium and low . . . 46
Figure 4.11 Distribution of Negative Binomial growth rate across all clusters with a usable fit . . . 47
Figure 4.12 Distribution of Mann-Kendall tau statistic across all clusters . . . 50
Figure 4.13 Number of clusters per Mann-Kendall trend direction, increasing, decreasing and no trend detected . . . 50
Figure 4.14 Trend category against Negative Binomial regression status, shown as a heatmap . . . 51
Figure 4.15 Trend confidence tier against Negative Binomial regression status, shown as a heatmap . . . 52

Figure 4.16 Distribution of coefficient of variation across all clusters 53
Figure 4.17 Distribution of year over year volatility across all clusters 53
Figure 4.18 Coefficient of variation plotted against year over year volatility, per cluster 54
Figure 4.19 Yearly item counts for the 6 clusters with the highest coefficient of variation 55
Figure 4.20 Yearly item counts for the 6 clusters with the lowest coefficient of variation 55
Figure 4.21 Yearly item counts for the 6 clusters with the highest year over year volatility 56
Figure 4.22 Yearly item counts for the 6 clusters with the lowest year over year volatility 56
Figure 4.23 Number of significant clusters per test before and after Benjamini-Hochberg correction 57
Figure 4.24 Yearly item counts with linear and polynomial fits, clusters selected for strongest trend signal 58
Figure 4.25 Yearly item counts with linear and polynomial fits, clusters selected to illustrate classification uncertainty 59
Figure A.1 Distribution of item clusters by platform origin. Bars show the number of clusters containing items from YouTube only, Stack Exchange only, or both platforms, out of 2470 total clusters 76
Figure A.2 Distribution of linear regression and Mann-Kendall p-values across all clusters 77
Figure A.3 Distribution of Negative Binomial regression p-values across all clusters with a usable fit 77

List of Tables

Table 3.1 Clustering Hyperparameters 19
Table 3.2 Terminology used for the clustering stage and beyond 22
Table 4.1 Non-noise items and item cluster assignments per year 34
Table 4.2 Manual extraction accuracy by platform 35
Table 4.3 Extraction failure and discard rates by platform 35
Table 4.4 Item character length (LLM input message), successful versus discarded, by platform 36
Table 4.5 Topics per item and cap hit rate, by platform 36
Table 4.6 Clustering scorecard 37

Table 4.7 Manual cluster coherence review 40
Table 4.8 Top 10 item clusters by size 42
Table 4.9 Bottom 10 item clusters by size 43
Table 4.10 Top 10 clusters by Negative Binomial growth rate 48
Table 4.11 Bottom 10 clusters by Negative Binomial growth rate 49
Table 4.12 Case study clusters, strongest trend signal 58
Table 4.13 Case study clusters, classification uncertainty 59
Table A.1 List of YouTube channel handles included in the data collection corpus 73

Listings

3.1 Prompt used for Stack Exchange sources 23
3.2 Prompt used for YouTube sources 24

Acronyms

LDA Latent Dirichlet Allocation
LLM Large Language Model
NLP Natural Language Processing
TF-IDF Term Frequency–Inverse Document Frequency
c-TF-IDF Class-based Term Frequency-Inverse Document Frequency
HDBSCAN Hierarchical Density-Based Spatial Clustering of Applications with Noise
DBSCAN Density-Based Spatial Clustering of Applications with Noise
PCA Principal Component Analysis
t-SNE t-distributed Stochastic Neighbor Embedding
UMAP Uniform Manifold Approximation and Projection
DBCV Density-Based Clustering Validation
AI Artificial Intelligence

1

Introduction

Software Engineering research and industry practice do not always align well. Academic work focuses on generalizable knowledge and long-term contributions, software practitioners are actually driven by immediate problem resolution and economic revenue on a day-to-day basis [9, 77]. This mismatch, often called the research-practice gap, has been a recurring topic of discussion in the Software Engineering community for many decades [23, 27]. These differences may result in varying emphases between topics discussed in academic publications and the topics that are relevant to industry practitioners. Understanding this gap requires some way of observing what practitioners actually care about, rather than only looking at citation counts or conference programs.

There are some places to look for the interests of practitioners in the field of Software Engineering, the different online media platforms in which practitioners discuss and comment a variety of topics, these platforms represent a growing source of informal discussion about software development practices, tools, and emerging trends. Stack Exchange sites [19], such as Stack Overflow, capture concrete problems and their solutions in the practitioner's own words, and have already been used as a data source in Software Engineering research in order to understand the popular trends [3]. YouTube, on the other hand, hosts conference talks, tutorials, informal videos and podcasts that reveal not just technical content, but also what practitioners choose to talk about and how they frame it. YouTube has been used in some Software Engineering research studies in order to map educational and training content [53, 64]. In this sense, these studies have provided an overview of the type of content that has been found in the platform regarding educational Software Engineering.

Compared to academic publications, these sources are large, noisy, constantly updated and not filtered by peer review, which makes them harder to analyze but arguably more representative of the discourse that actually surrounds day-to-day software development for practitioners.

Extracting structured insight from this kind of informal, high-volume text has historically been labor-intensive. Traditional topic modeling approaches, most notably Latent Dirichlet Allocation (LDA) [5], have been used to identify recurring themes in developer discussions, for example on Stack Overflow [3, 63]. However, these statistical methods like LDA require the number of topics to find to be chosen in advance and fail to capture semantics of a text [38]. The recent emergence of capable LLMs in a variety of natural language tasks [11, 49, 76] offers a different approach to understand and extract the topics of a text without human intervention. An LLM can be directly prompted to extract a human-readable topic label from

a given piece of text [49]. This shifts much of the manual effort from tuning a statistical model to designing a good prompt [21], and produces output that is already interpretable without further post-processing [73].

This thesis investigates whether such an LLM-based approach can be used to extract topics from informal Software Engineering discourse at scale, and whether the resulting topics reveal any meaningful patterns in how practitioner attention shifts over time. We build a pipeline that collects data from YouTube and Stack Exchange, uses an LLM with specific prompts to extract a list of topic labels for each item, groups semantically similar topic labels into clusters using a small sentence embedding model and density-based clustering, and analyzes how the prevalence of each cluster changes over a multi-year window.

One challenge in this last step is that trends in discourse volume over time rarely behave in a simple way. A topic's popularity can grow steadily, spike and then plateau, or reverse direction entirely, and several topics could have years with no recorded occurrences at all. A standard method such as linear regression is not built for this: it assumes a straight line relationship that many of these patterns violate, and it breaks down once a series contains true zero counts. We therefore rely on two methods suited to short series with real zeros, the Mann-Kendall trend test [36, 41], a non-parametric test that checks for a monotonic trend without assuming linearity or a specific data distribution, and Negative Binomial regression [32], a regression model built for count data that treats a year with zero occurrences as a normal outcome rather than an edge case, yielding an estimated annual growth rate together with a significance test. Combining the two lets us classify each topic's development as emerging, stable, or declining, with a confidence level that reflects whether both tests agree.

1.1 Goal of this Thesis

The objective of this work is to design, implement, and evaluate a pipeline that extracts and analyzes topics from informal Software Engineering discourse, and to use it to describe how practitioner attention to different Software Engineering topics has shifted over a defined time window of seven years (2020-2026). Along the way, this thesis also looks at whether LLM-based topic extraction is a viable approach for this kind of data.

1.2 Overview

In addition to the present chapter, which is Chapter 1, Chapter 2 introduces the background needed to follow the remainder of the thesis. Chapter 3 describes the methodology behind the data collection, topic extraction, and clustering pipeline. Chapter 4 presents the results, followed by a discussion of what they mean and the threats to their validity. Chapter 5, situates this work within the related literature, and Chapter 6 concludes.

2

Background

This chapter introduces the concepts required to fully understand this thesis. We first discuss the research-practice gap and the role of informal discourse as a lens to practitioner concerns. We then introduce the core Natural Language Processing (NLP) and machine learning building blocks used in the pipeline like prompt engineering, sentence embeddings, and density-based clustering, before turning to the statistical methods used to characterize topic trends over time.

The Research-Practice Gap in Software Engineering

The term research-practice gap refers to the observed divergence between the topics, methods and priorities pursued by academic Software Engineering research and those that are relevant to industry practitioners [23]. Several studies have attempted to characterize this gap by comparing the content of academic publications against surveys or interviews with practitioners [4, 22, 39]. Another approach to characterize the gap is by mining artifacts such as developer forums or software repositories to infer what practitioners actually work on [3, 62]. A persistent challenge in this line of work is scale: manually coding practitioner discourse into topics is time consuming and difficult to reproduce across large corpora or over long time windows, which historically limited such studies to comparatively small samples.

Informal Practitioner Platforms as Data Source

Gray literature refers to materials that are produced outside of traditional academic publishing channels and are not formally peer-reviewed. According to existing definitions, gray literature includes documents such as technical reports, white papers, blog posts, industry guidelines, and other materials created by practitioners or organizations [40, 52, 72]. These sources are typically published through non-commercial or informal channels and may not follow strict academic quality control processes [26, 61]. In Software Engineering research, gray literature has gained importance because it often reflects practical experiences and emerging industry trends that are not yet covered in academic publications [24, 26, 70]. Researchers have argued that including gray literature can provide a more complete understanding of real-world software development practices. However, the use of gray literature also introduces challenges, such as variability in quality, lack of peer review, and potential bias [54]. For this reason, methodological guidelines emphasize the need for careful

documentation, transparency, and quality assessment when such sources are included in research studies [25]. Stack Exchange platforms provide structured question-and-answer data annotated with community-driven signals such as votes and view counts, making them a frequently used source in Software Engineering mining studies [3, 63]. YouTube, by contrast, provides long-form, less structured content such as conference talks, tutorials and podcasts that capture practitioner discourse in a different form. While YouTube has been used as a data source in a different Software Engineering studies, for instance to map educational and training content [53, 64], its use for large-scale analysis of practitioner discourse remains limited.

Topic Extraction with LLMs and Topic Modelling

Topic modeling refers to machine learning techniques for identifying themes or topic discussed within large text corpora. Traditional topic extraction methods were primarily based on probabilistic models such as LDA [5]. LDA defines each text/document as a probability distribution over topics, and each topic is defined by a probability distribution over words. LDA has shown great results in topic extraction, however LDA fails to capture the semantic meaning of texts/documents [38, 63].

An alternative to topic extraction using classical statistical methods like LDA is to use deep learning models based on the transformer architecture to extract topics from a given text [65]. Transformers use self-attention mechanisms to model contextual relationships between words within a text, allowing them to capture meaning beyond simple co-occurrence statistics. Large Language Models, such as GPT-3 [6] and LLaMA [28] which are built on this architecture and have demonstrated strong capabilities in representing semantic relationships and understanding nuanced language use [11, 76]. LLMs are better suited for analyzing informal and conversational data [38], such as podcasts or online discussions, where vocabulary may vary and expressions are less structured.

Prompt Engineering

In order to extract topics from a text using a Large Language Model, the model must be instructed to perform that task, which is done entirely with natural language instructions. The manner in which a task is phrased has a substantial effect on the quality, consistency, and structure of the model's output. Prompt engineering refers to the practice of systematically designing these instructions to reliably elicit a desired kind of output from a language model [21, 74].

A range of established techniques exist for this purpose [58, 60]. Explicitly instructing an output format constraint for instance, instructing the model to respond only with a JSON object conforming to a specified schema [73]. Few-shot prompting, in which the prompt includes one or more examples about the task can help steer the model toward a desired granularity or style [60]. Role or context framing, in which the prompt establishes the perspective or domain the model should adopt (e.g., "you are analyzing a software engineering discussion"), can further improve the relevance and specificity of extracted topics [10].

Prompt design becomes especially important in settings, such as this thesis, where a single extraction task must be applied across different data sources. A prompt tuned for short, technically dense Stack Exchange posts may perform poorly on long, conversational YouTube transcripts, and vice versa. Consequently, source specific prompt variants, rather than a single instruction, are typically required to obtain comparable extraction quality across sources.

Sentence Embeddings and Semantic Similarity

Once topics have been extracted as short text labels, semantically equivalent or closely related topics (e.g., "memory leak" and "memory management issue") must be identified and grouped together. This requires a way to quantify the semantic similarity between two pieces of text.

One classical approach to representing text numerically is Term Frequency–Inverse Document Frequency (TF-IDF), which represents documents as sparse vectors weighted by word frequency and rarity across a corpus [57]. However, TF-IDF based representations capture only surface-level lexical overlap and fail to recognize semantic similarity between topics phrased using different words [34]. This thesis therefore does not rely on TF-IDF vectorization. Instead, we use a small pretrained sentence embedding model to map each extracted topic label into a fixed-dimensional continuous vector space such that semantically similar inputs are mapped to nearby vectors, regardless of lexical overlap. Models from the Sentence-Transformers family are trained specifically to produce embeddings that are effective for downstream similarity computations, such as cosine similarity between vectors [56]. In particular, we use all-MiniLM-L6-v2, a compact transformer model obtained through knowledge distillation [67].

Dimensionality Reduction

The sentence embeddings produced by an embedding model are high dimensional vectors, and clustering algorithms that rely on distance estimates, such as HDBSCAN [8], become less reliable as dimensionality grows, in high dimensional spaces, distances between points tend to converge toward similar values, making it difficult for a density based method to distinguish dense regions from sparse ones [46]. Dimensionality reduction techniques address this by mapping the original vectors into a lower dimensional space while attempting to preserve the structure of the original data.

Classical linear techniques such as Principal Component Analysis (PCA) reduce dimensionality by projecting the data onto the directions of greatest variance [29], but this linear assumption can fail to preserve the local neighborhood structure that density based clustering depends on, particularly when the semantically meaningful structure of the embeddings is not aligned with the directions of maximum variance [15]. Non linear techniques such as t-distributed Stochastic Neighbor Embedding (t-SNE) preserve local neighborhood structure more faithfully [2], but are primarily designed for producing two or three dimensional visualizations and do not scale as well to reducing many points down to several dozen dimensions for downstream clustering [68].

UMAP is a non linear dimensionality reduction technique that first constructs a graph representation of the neighborhood relationships between points in the original high dimensional space, and then optimizes a lower dimensional layout intended to preserve this neighborhood structure as closely as possible [44]. Compared to t-SNE, UMAP is reported to preserve more of the global structure of the data in addition to local neighborhoods, and to scale more efficiently both to larger datasets and to a moderate number of target dimensions, which makes it a common preprocessing step before density based clustering [43]. A key parameter of the algorithm is the number of neighbors considered when constructing the neighborhood graph, which controls the balance between preserving fine grained local structure and broader global structure in the reduced representation.

Density-Based Clustering with HDBSCAN

Given embedded topic vectors, the next step is to group them into clusters representing broader thematic categories. Unlike centroid-based methods such as k-means clustering, which require the number of clusters to be specified in advance and assume roughly spherical, similarly sized clusters, density-based clustering methods identify clusters as regions of high point density separated by regions of lower density, without requiring the number of clusters to be known beforehand.

HDBSCAN extends the classical Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [16] algorithm by building a hierarchy of clusters across varying density thresholds and then extracting the most stable clusters from this hierarchy [8]. Rather than relying on a single, global density threshold as in DBSCAN, HDBSCAN computes a mutual reachability distance between points and constructs a hierarchy that captures clustering structure at multiple density levels simultaneously, clusters are then selected from this hierarchy based on their stability across density thresholds. A key parameter of the algorithm is the minimum cluster size, which determines the smallest grouping of points that will be considered a genuine cluster rather than noise.

This makes HDBSCAN particularly well suited to topic clustering, where cluster sizes and densities can vary substantially, some software engineering topics are discussed far more densely than others and not every extracted topic label necessarily belongs to a meaningful cluster. Unlike k-means, HDBSCAN explicitly supports labeling low-density or outlying points as noise rather than forcing every point into some cluster, which is important given that LLM-extracted topic labels may occasionally be idiosyncratic, overly specific, or otherwise not representative of a broader recurring theme.

Cluster Validity

Assessing the quality of a clustering solution is inherently difficult because, unlike in supervised settings, there is no ground truth assignment of items to topics to compare against. In the absence of such a reference, cluster validity measures instead rely only on the geometry of the data itself, typically by quantifying how compact each cluster is internally and how well separated it is from other clusters, and combining these two quantities into a single numerical score.

Popular measures such as the Silhouette index or the Davies Bouldin index follow this approach, but do so by comparing distances to cluster centroids, which implicitly assumes that clusters are convex and roughly similar in size and density [35]. This assumption does not hold for density based methods such as HDBSCAN, which by design can produce clusters of arbitrary shape and substantially different sizes, and which can also leave many points unassigned as noise. Applying a centroid based validity measure to such a clustering solution can therefore produce misleading scores, since the measure penalizes exactly the kind of shape and size variation that HDBSCAN is designed to capture.

DBCV is a validity index designed specifically to address this mismatch, by evaluating density based clustering results on their own terms rather than assuming convexity [48]. Rather than measuring distances to a single cluster centroid, DBCV estimates, for every cluster, a measure of its internal density using a construction called mutual reachability distance, the same distance notion used internally by HDBSCAN itself, and then compares this internal density against the density separating the cluster from its nearest neighboring cluster. Clusters that are internally dense and well separated from their neighbors contribute positively to the overall score, while clusters that are sparse or poorly separated contribute negatively. These per cluster contributions are combined, weighted by relative cluster size, into a single score for the entire clustering solution, ranging from negative one to positive one, with higher values indicating a solution that better recovers dense, well separated groups.

DBCV provides a single, corpus wide measure of how well the clustering procedure overall separates meaningful groups from noise and from each other.

Statistical Trend Analysis

The final analytical stage of this thesis concerns detecting and characterizing trends in cluster prevalence over successive years. Four complementary methods are used, each capturing a different aspect of trend behavior.

Linear Regression. Fits a straight line to the observed data points, yielding a slope that indicates the average rate of change over time. It is simple and highly interpretable, but assumes a constant rate of change throughout the observation window, which is often unrealistic for topics that accelerate, plateau, or reverse direction.

Polynomial Regression (Degree 2). Fitting a second-degree polynomial (a parabola) to the data relaxes the linearity assumption while remaining low-complexity and easy to interpret. The sign of the leading (quadratic) coefficient indicates the curvature of the trend, concave-up or concave-down, revealing acceleration, deceleration, or reversal patterns invisible to the single slope value produced by linear regression.

Mann-Kendall Trend Test. The Mann-Kendall test is a non-parametric, rank-based statistical test for detecting a monotonic trend in a time series, without assuming linearity, normality, or a specific functional form [36, 41]. It compares the relative ordering of all pairs of observations to compute Kendall's rank correlation coefficient (τ), which indicates

the direction and strength of the trend, along with an associated p-value indicating its statistical significance. Because it relies only on the relative ranking of observations rather than their exact values, the Mann-Kendall test is highly robust when it comes to outliers and to violations of distributional assumptions. It is important to note, however, that with a small number of 7 yearly observations per cluster, the statistical power of the test is limited, and its results must be interpreted with appropriate caution.

Negative Binomial Regression. Negative Binomial regression is a count based regression model used to detect trends in time series whose values are non-negative event counts, including years in which no occurrences were observed at all [13]. It models the expected count at each time point as a function of time, while additionally estimating a dispersion parameter that allows the variance to exceed what a simpler count model (a Poisson process) would predict, accounting for the fact that observed count sequences are typically noisier than a strict random occurrence assumption would imply. The resulting slope coefficient yields an estimated average growth rate per year, together with a p-value indicating whether the trend is statistically distinguishable from no change. Unlike linear, polynomial, or rank based methods, the Negative Binomial regression treats a zero observation as a normal, expected outcome rather than an edge case, making it particularly well suited to series containing years with no observed occurrences, a situation the other three methods handle poorly or not at all

Used together, these four methods provide complementary evidence: linear regression establishes a simple interpretable baseline, polynomial regression reveals curvature and reversal patterns invisible to a straight-line fit, the Mann-Kendall test provides a distribution-free assessment of whether an observed directional trend is statistically defensible given the available data, and Negative Binomial regression offers a count-based estimate of trend that, unlike the other three methods, does not require a series to be free of zero-occurrence years

3

Methodology

This chapter presents the methodology designed to extract, group, and analyze Software Engineering topics from informal practitioner discourse. We start by presenting the research questions guiding this work, followed by their operationalization. We then detail how the underlying data is collected, how topics are extracted from this data with a Large Language Model, how the resulting topics are clustered, and finally, which statistical methods are applied to characterize how these topics develop over time.

3.1 Research Questions

This section contains all research questions and a detailed motivation and explanation of them.

Although prior work has applied automated text analysis to Software Engineering artifacts such as issue trackers and developer forums, there is still limited research on analyzing practitioner discussions across multiple informal platforms using Large Language Models. In particular, it is not yet well understood whether text analysis pipelines using LLMs can reliably extract meaningful Software Engineering topics from conversational and unstructured media such as podcasts, YouTube videos, or online discussions.

To address this gap, this thesis will investigate both the feasibility of automated topic extraction and the characteristics of the topics discussed by practitioners. This study is guided by the following research questions.

> **RQ1:** To what extent can an automated text analysis pipeline identify Software Engineering topics from different online platforms?

Before analyzing topic trends, it is necessary to determine whether the proposed pipeline is capable of extracting meaningful topics from informal textual sources. Unlike academic publications or structured documentation, practitioner media such as YouTube video transcripts and Stack Exchange threads often contain conversational language, incomplete sentences and noise.

This research question evaluates the feasibility of the proposed pipeline. The evaluation of RQ1 will focus on whether the generated topic labels correspond to meaningful Software Engineering concepts and whether the proposed clustering process successfully groups semantically similar labels into coherent topic clusters. Establishing the reliability of the

extraction process ensures that the subsequent analyses are based on a technically sound foundation.

> **RQ2:** What Software Engineering topics are most frequently discussed on the analyzed platforms, and how do these topics evolve over time?

After establishing that the pipeline can extract meaningful topics, the second research question focuses on identifying the dominant themes discussed within practitioner communities. By analyzing large collections of YouTube videos and Stack Exchange threads, this question asks which Software Engineering topics appear most frequently across these platforms, and how the prevalence of these topics changes over time. Answering this question provides empirical insight into how practitioner interest in the Software Engineering community evolves.

3.2 Operationalization

In order to answer the two research questions above, this thesis implements an automated text analysis pipeline that processes textual data from YouTube and Stack Exchange discussions related to Software Engineering. The pipeline extracts topics from every collected item. Then groups these topics into clusters and analyzes the frequency and evolution of these clusters over time. Sections 3.3 to 3.7 describe this pipeline in full technical detail, this section instead explains, at a higher level, how the output of that pipeline is used to answer each research question.

3.2.1 RQ1: Evaluating Pipeline Feasibility

To address **RQ1**, we evaluate two parts of the pipeline separately.

First, a random sample of 100 items per platform (YouTube and Stack Exchange) is manually inspected by the author to determine whether the extracted topic label accurately reflects the content of the corresponding video or discussion thread. Each inspected item is assigned a binary outcome, either **Accurate** or **Inaccurate**, based on this determination. This qualitative inspection gives an indication of whether the large language model used in the extraction step (Section 3.5) is capable of identifying meaningful topics from conversational and unstructured text. As this inspection is carried out by a single person, this is acknowledged as a limitation in Section 4.3.

Second, the clustering step described in Section 3.6 is evaluated using both a targeted sample and a global metric. We draw a random sample of 200 topic label clusters, the direct output of HDBSCAN before the item clusters described in Section 3.6.5 are built from them, and check whether the topic labels grouped into each sampled cluster are indeed semantically related. For this, we perform a manual review of the member labels of each sampled cluster.

In addition to this sample-based check, we report the DBCV [48] score for the full clustering solution, a validity index designed for density-based methods such as HDBSCAN, which

does not assume convex clusters or similar sized clusters the way metrics like other evaluation metrics do. While the sampled coherence checks assess whether specific clusters are internally sensible, DBCV provides a single, corpus-wide measure of how well the clustering as a whole separates the formed groups from noise. If the manual review that confirms coherence, and the overall DBCV score is favorable, this indicates that the clustering procedure consistently groups related labels together, and that the pipeline as a whole produces meaningful topics.

3.2.2 RQ2: Topic Frequency and Evolution

To address **RQ2**, the final item clusters produced by the pipeline (Section 3.6), each one operationalizing a single topic as defined in Section 3.6.5, are analyzed along several complementary dimensions, summarized below and defined precisely, together with their exact formulas in Section 3.7.

Topic size and yearly share. For every item cluster, which represents a topic as established in Section 3.6.5, we determine how prominent it is in two ways. The first is its size, the total number of item cluster assignments it received across the whole seven year window. The second is its yearly share, a separate value computed for each year on its own. A cluster's share for a given year is not a proportion of the cluster's own size. It is the proportion of that entire year's item cluster assignments, counted across every cluster, that happen to belong to this one cluster. For example, if a cluster received 40 item cluster assignments in 2023, and 500 item cluster assignments were made in total across all clusters that year, the cluster's share for 2023 is 40 divided by 500, or 8 percent. So while a cluster has one size value for the whole window, it has seven separate share values, one per year, and these seven values are what let us see whether the cluster's relative prominence grew, shrank, or stayed put from one year to the next.

We use each cluster's yearly share, rather than its raw yearly count, as the basis for the diversity and trend analyses described below. The total amount of data collected is not the same across the seven years under investigation, so a raw count could make a topic look more popular in a given year simply because more data was collected that year overall. Share corrects for this by measuring a cluster against the total volume of that same year, rather than against a fixed number.

Diversity across topics. To capture not just how a single topic behaves but how discussion is distributed across all topics in a given year, we additionally compute the Shannon entropy of the yearly share distribution. This measure is low when discussion is concentrated on a small number of dominant topics and high when it is spread evenly across many, complementing the per topic analysis with a view of the corpus as a whole.

Detecting trends over time. Since a relatively short time series of only seven points $(2020 - 2026)$ can behave in ways that a single method may fail to capture, we combine several measures to capture more information about the possible trends.

We compute two descriptive statistics for each topic's yearly share: an ordinary least squares regression, whose slope indicates the rate of change per year assumed by the best fitting straight line, and a polynomial regression of degree two, which can reveal whether a trend is accelerating, decelerating, or reversing direction, something a straight line simply cannot show. Separately, we estimate each topic's growth rate using Negative Binomial regression, applied to the raw yearly item counts rather than the share. We use this method because item counts are integers that can genuinely be zero in a given year and the continuous data methods above do not handle such zero counts well, whereas Negative Binomial regression is built specifically to model this kind of count data, including years without any items at all. We also apply the Mann-Kendall test, which is able to identify a consistent monotonic trend with the topic's yearly time series, without assuming the trend follows any particular shape. The Mann-Kendall test and the Negative Binomial growth rate are combined into a single classification for each topic, indicating whether it is emerging, stable or declining, together with a confidence level based on whether both methods agree and are individually statistically significant. The linear and polynomial fits remain useful descriptive summaries but do not directly determine this classification.

Summarizing the whole dataset. Beyond describing individual topics, we also summarize the dataset as a whole. For every measure described above, we report descriptive statistics, including the mean, median, standard deviation, and percentiles, across all clusters, giving an overall picture of how size, growth, and variability are distributed across the full set of topics rather than relying only on individual examples. We also rank clusters along each of these measures, for example by size or by growth rate, and report the clusters at the top and bottom of each ranking, to highlight both the most prominent topics overall and the topics whose prominence changed the most over the seven year period. Two additional measures, the coefficient of variation and the year over year volatility of a topic's share, are reported to distinguish topics whose share changes smoothly and consistently from those whose share fluctuates a great deal without a clear direction.

3.3 Overview of the Pipeline

To answer these two research questions, we built a pipeline made up of four stages: (1) data collection, (2) topic extraction with an LLM, (3) embedding and clustering of the extracted topics, and (4) statistical frequency and trend analysis. Figure 3.1 shows a visual overview of the four pipeline stages, from raw data collection to trend analysis, with the output of one stage feeding into the input of the next. The following sections describe each stage in detail, in the order in which they are executed.

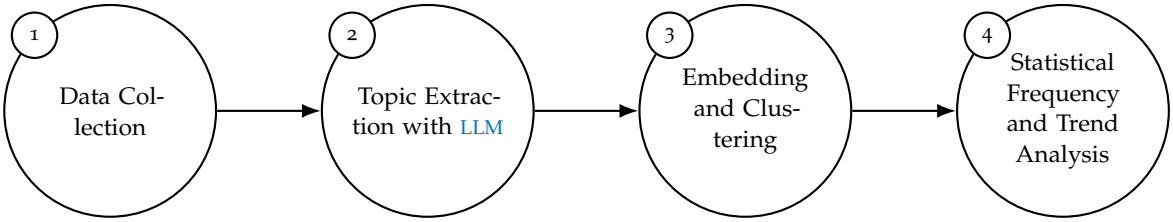


Figure 3.1: Overview of the four pipeline stages

One terminological point is worth flagging before these stages are described in detail. The third stage, clustering, produces two related but distinct groupings: a grouping of the extracted topic labels themselves, and, derived from it, a grouping of the underlying items (YouTube videos or Stack Exchange posts) that contributed those labels. This thesis calls the first a **topic label cluster** and the second an **item cluster**. The two are closely related, every topic label cluster has exactly one item cluster built from it, but they are not interchangeable: a topic label cluster's size counts labels, while an item cluster's size counts items. Section 3.6.5 defines this distinction precisely. It is introduced here, immediately after the Operationalization section above, so that the word "cluster," used informally both in that section and throughout the pipeline description that follows, can be read with this split in mind.

3.4 Data Collection

We collect data from two sources: Stack Exchange and YouTube. Both sources are collected independently but are processed by the same downstream pipeline (topic extraction, clustering, frequency and trend analysis), which allows us to compare a structured question-and-answer platform with a long-form, conversational platform using the same methodology.

3.4.1 Stack Exchange Data

We collect posts from the Stack Overflow [20] and Software Engineering [17] sites using the Stack Exchange Web API [18]. For each post, we retrieve the title, the body text, the creation date, and the associated tags. We restrict the collection to posts created between 2020 and 2026, since this is the time window used in the frequency and trend analysis (Section 3.7).

Before a post is passed to the topic extraction stage, we apply the following preprocessing: We remove HTML tags and code blocks from the post body, keeping only the natural language text, since code snippets are not useful input for topic extraction and can confuse the LLM.

3.4.2 YouTube Data

We collect videos using the YouTube Data API [75] from a list of 107 YouTube Channels (see Table A.1 for full list of channels), restricted to videos published between 2020 and 2026. For each video, we retrieve the title, description, and publication date and we download the transcript using the youtube-transcript-api python package [78]. We only used videos that are publicly available in YouTube, that are in English, have a minimum duration of 5 minutes and have an English transcript of the content of the video.

3.5 Topic Extraction

The goal of this stage is to extract the topics that are discussed in each item, a Stack Exchange post or a YouTube video transcript. Each item may have multiple topics present in the text, and the output of this stage, a list of topic labels per item, is the input to the embedding and clustering stage described in Section 3.6. Because the results of this thesis, cluster composition, topic frequency, and trend statistics, are built on top of these extracted topic labels, the design choices described below directly determine how far the final results can be trusted. Figure 3.2 shows a concrete example of this stage's input and output before the remainder of this section describes how it is produced.

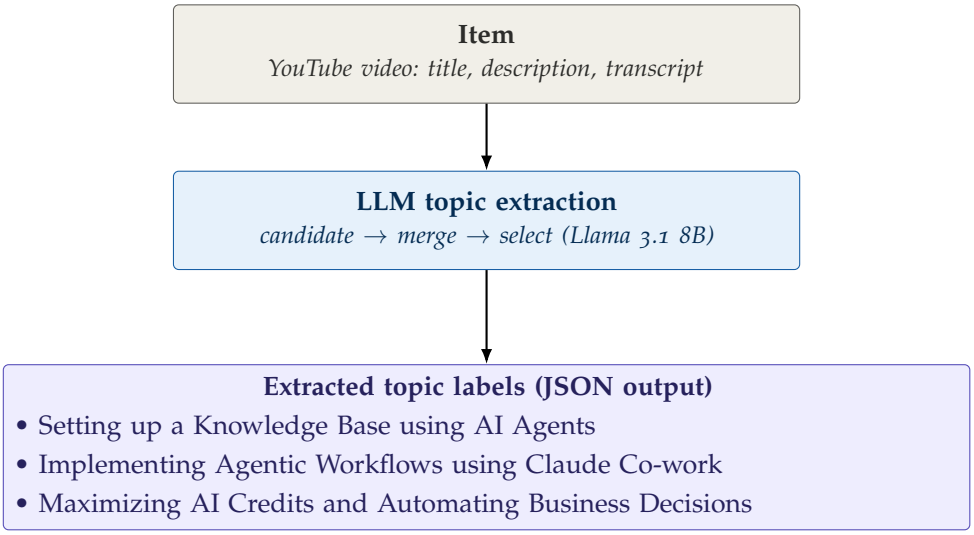


Figure 3.2: A concrete example of the topic extraction step: an item's text is passed to the LLM, which returns a JSON list of extracted topic labels.

3.5.1 Model and Setup

We use **llama3.1:8b** [45], a large language model with approximately eight billion parameters, hosted by the Software Engineering Chair of the Saarland University and accessed through an endpoint compatible with the OpenAI chat completions API (Open WebUI). We use a generation temperature of 0.2 to keep the output reasonably consistent across repeated runs on similar input text, while still allowing enough variation for the model to phrase topics naturally.

3.5.2 Data Sources and LLM Input Construction

The two platforms that are investigated in the thesis differ substantially in structure, a different input message is assembled for the Large Language Model depending on the source, even though both messages are sent to the same model with the same settings.

For a YouTube video, the message is built from the video title, its description, and its full transcript, concatenated as labelled sections and separated by blank lines. Any of these three fields that is missing or empty is simply left out of the message rather than included as an empty section. Importantly, the complete transcript is passed to the model so that the model reads the entire spoken content of the video, not a shortened summary of it.

For a Stack Exchange post, the input message is built from the question title, its tags joined into a single comma separated string, the body of the question itself, the body of the accepted answer, and the body of every additional answer attached to the question. Tags are explicitly framed to the model as context clues about the domain of the question rather than as topics to extract directly, which is intended to prevent metadata that is not actually discussed in the text from inflating the topic count (see Section 3.5.3).

3.5.3 Prompt Design

We use two prompts, one for Stack Exchange posts and one for YouTube video transcripts, reflecting the difference in structure and typical length between the two sources. Both prompts combine three prompting techniques established in the prompt engineering literature.

The first technique is role framing [10], both prompts open by instructing the model to act as an expert Software Engineering researcher. Framing the model this way is reported to improve the relevance and vocabulary of the generated text for domain specific tasks.

The second is structured output [73], both prompts require the model to return a single JSON object containing only the requested fields, with no prose, no markdown fences, and no explanation around it. This is what allows the responses to be parsed automatically without manual intervention for each one.

The third is an explicit step by step task decomposition, a form of instructed reasoning in which the model is walked through three ordered steps rather than asked to produce the final answer directly. This decomposition approach follows the same general principle used by Montes et al. [47] for prompting large language models to perform phases of thematic analysis in software engineering research, where a qualitative task is broken into named, ordered stages within a single prompt rather than left as one open instruction. In Step 1 the model lists every candidate topic it notices in the text. In Step 2 it merges topics from Step 1 that describe the same underlying idea, even if they were mentioned in different parts of the source text or phrased differently. In Step 3 it selects the final topics from the merged list, keeping between one and eight topics for YouTube or one and three topics for Stack Exchange, and prioritising the topics most central to the item rather than trying to be exhaustive. Each step in the prompt is accompanied by a worked example of the intended granularity, for instance showing that "Setting Up a CI/CD Pipeline with GitHub Actions" and "Automating Deployments Using GitHub Actions" describe the same idea and should be merged into a single topic rather than kept as two. Figure 3.3 walks through this same example across all three steps.

The complete system prompts used for each source are given in Listing 3.1 (Stack Exchange) and Listing 3.2 (YouTube).

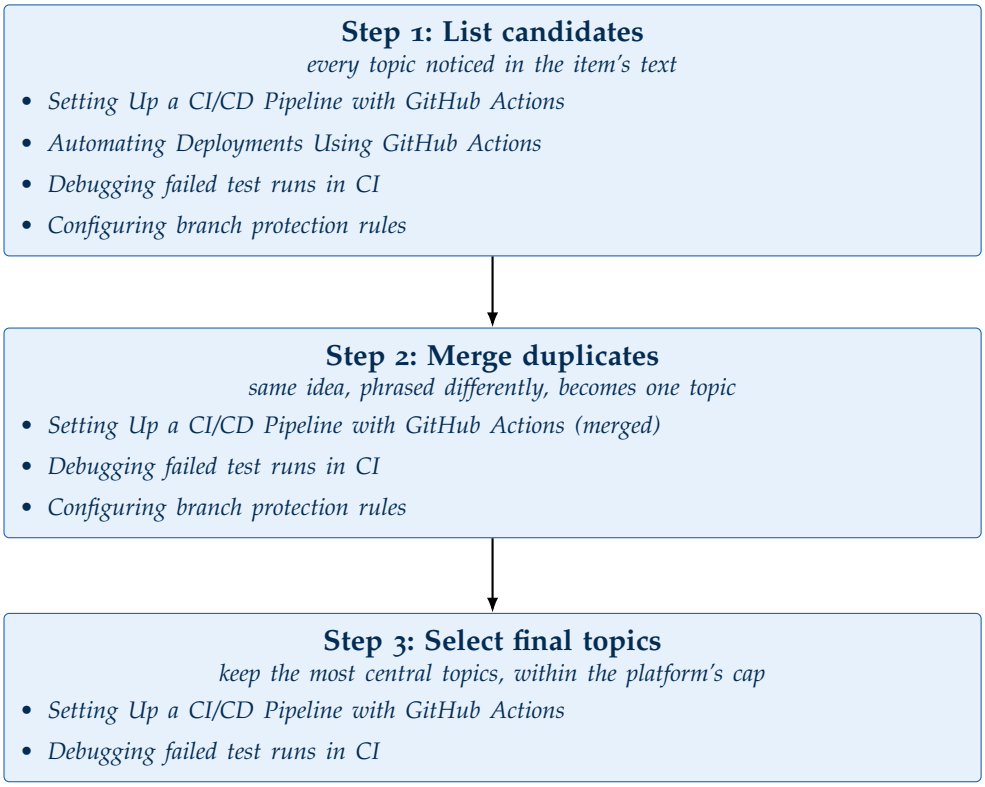


Figure 3.3: The three-step task decomposition used in both prompts, applied to a small worked example: candidate topics are noticed, duplicates are merged, and the least central remaining topic is dropped at selection.

3.5.4 Iterative Prompt Refinement

The prompts described in Section 3.5.3 are the result of one round of revision rather than the first version we used. An earlier version of both prompts, asked the model to return a topic list directly, in a single pass, without the candidate, merge, and select steps described above, and without a limit on how many topics could be returned.

In practice, this earlier version tended to produce a long, uncapped, and often repetitive list of topics per item. For YouTube specifically, a typical video could return on the order of forty topics, many of them near duplicates of each other or only loosely relevant to the video's actual content. Introducing the capped, three step decomposition described in Section 3.5.3 reduced this to a small, curated set of topics per item, and is the version that produced the data analyzed in this thesis. We treat this change as a form of internal prompt validation, before settling on the final prompt, we observed that the uncapped, single step version was not producing topic labels of a usable granularity, and revised the prompt accordingly.

3.5.5 Output Format and Post-processing

The model is instructed to return its answer as a JSON object, as described in Section 3.5.3, but instruction following is not guaranteed, so the raw response returned by the model is passed through a dedicated parsing routine before it is accepted. This routine strips markdown code fences if present, locates the first opening brace or bracket in the response to skip any leading text the model may have added despite the instructions, finds the matching closing character to discard any trailing text after it, and falls back to accepting a plain JSON array if no JSON object is found, which covers responses still following the format used by the earlier prompt version described in Section 3.5.4.

This process should be understood as a defensive layer that compensates for imperfect instruction following on the part of the model, not as evidence that the model reliably produces well formed output on its own.

A response is treated as an extraction failure once it cannot be parsed as valid JSON by this routine or once it parses but is missing the expected topics field entirely.

3.6 Embedding and Clustering of Topic Labels

Once every item has a list of extracted topic labels that represent the topics that are discussed in the item, we group labels that are semantically similar together, so that labels such as "memory leak" and "memory management issue" end up in the same cluster even though they are syntactically different but semantically equivalent. This grouping happens in three stages: each unique topic label is first turned into a numeric vector using a sentence embedding model, that vector is then reduced to fewer dimensions, and the reduced vectors are finally clustered with the HDBSCAN algorithm. The remainder of this section describes each of these stages in detail.

3.6.1 Normalization of Topic Labels

Before a topic label is embedded, it is passed through a normalization step. This step exists to remove trivial differences in writing, such as case or punctuation, so that two labels that are effectively identical are not accidentally treated as different simply because they were written slightly differently by the LLM. Concretely, each label is converted to a canonical unicode form, lowercased, stripped of punctuation, and lemmatized to reduce inflected words to a common base form, so that, for example, Testing and Test are treated as the same underlying word.

3.6.2 Sentence Embeddings

We embed every unique topic label using the pretrained sentence embedding model all-MiniLM-L6-v2 from the Sentence-Transformers family [33, 56]. This model maps each topic label to a fixed 384-dimensional vector, such that labels with similar meaning are mapped to nearby vectors. We chose this model over a classical TF-IDF representation because it is able to capture semantic meaning much better [51]

3.6.3 Dimensionality Reduction with UMAP

The vectors produced in the previous step have 384 dimensions each. Clustering directly on vectors with that many dimensions is problematic because HDBSCAN, relies on distances between points to estimate how dense a region is, and those distance estimates become unreliable as the number of dimensions grows. To address this, we reduce every embedding to 16 dimensions using UMAP [44] before clustering, and it is this reduced representation, not the original 384 dimensional vector, that HDBSCAN actually clusters on.

Beyond the target number of dimensions, UMAP also depends on a number of neighbors parameter, which controls how many nearby points it looks at around each embedding when learning the shape of the data before compressing it. A small neighbor count lets UMAP follow fine, local detail in the data, at the cost of potentially fragmenting structure that only becomes visible when looking at a wider neighborhood. A large neighbor count favors this broader structure instead, at the cost of local detail. Table 3.1 reports the value used, and Section 3.6.4.1 describes how it was selected.

3.6.4 Clustering with HDBSCAN

We cluster the reduced embeddings using the HDBSCAN algorithm, with a minimum cluster size of 11 and Euclidean distance as the distance metric. Alongside minimum cluster size, which sets the smallest number of topic labels allowed to form a cluster on its own, HDBSCAN also depends on a minimum samples parameter, which controls how conservative the algorithm is when deciding whether a region of the embedding space is dense enough to belong to a cluster at all. A higher value requires a point to have more neighbors packed closely around it before that point is trusted as part of a dense region, which tends to produce fewer, more tightly packed clusters, at the cost of pushing more borderline points

into the noise cluster described below. We use a minimum number of samples of 16. Table 3.1 shows the HDBSCAN hyperparameters used, and Section 3.6.4.1 describes how these values were selected.

The clustering algorithm assigns every embedded topic label either to one of the discovered clusters or to a special noise cluster, meaning the label does not belong to any sufficiently dense group. We exclude topic labels in the noise cluster from the frequency and trend analysis in Section 3.7, since a cluster made up of noise points does not represent a coherent Software Engineering topic.

For every topic label that is assigned to a cluster, HDBSCAN additionally reports a membership probability, a value between 0 and 1 indicating how confidently that particular label belongs to its assigned cluster, rather than merely reporting that it does. A label sitting deep within a cluster's dense core receives a value close to 1, while a label sitting near the outer edge of a cluster, close to the boundary where the cluster's density starts to thin out, receives a lower value, even though both labels are formally counted as members of the same cluster. This value is what we mean whenever this thesis refers to a cluster's **mean membership probability**, the average of this per label value across every topic label assigned to that cluster, used throughout the remainder of this section and in Chapter 4 as a measure of how confidently, on average, a cluster's members were assigned to it, complementing DBCV, which measures the separation and density of the cluster as a whole rather than the confidence of its individual members.

Table 3.1: Clustering Hyperparameters

Algorithm	Parameter	Value
UMAP	Number of dimensions	16
UMAP	Number of neighbors	6
HDBSCAN	Minimum cluster size	11
HDBSCAN	Minimum samples of a dense region	16

3.6.4.1 Hyperparameter Selection via Grid Search

The four hyperparameters governing this stage (UMAP's number of components and neighbors, HDBSCAN's minimum cluster size and minimum samples of a dense regions) were selected through a grid search rather than fixed arbitrarily. The search jointly considered three criteria that are often in tension: cluster validity score (DBCV), assignment confidence (mean membership probability of a point belonging to a cluster), and cluster granularity, preferring fewer, larger, more generalizable clusters over many small fragments. All runs used a fixed seed (42), and since UMAP's random state is fixed, each configuration was evaluated once rather than averaged, as the result is deterministic. Each grid search run was evaluated on a stratified sample of 35,000 topic labels rather than the full corpus, with a new random sample of the same size drawn independently for every run, so the DBCV and noise ratio values reported in this section reflect this subset and are not directly comparable to the final clustering results reported in Chapter 4.

The search proceeded across three rounds, each narrowing the space based on the previous round's findings.

First round. Starting from a broad grid over all four hyperparameters, this round was used to see which hyperparameters mattered most for each of the three criteria, and whether a single configuration could simply maximize all three at once. Neighbor count and minimum samples both showed a strong negative relationship with DBCV, higher values of either tended to lower validity. Of the two HDBSCAN parameters, minimum cluster size was the strongest driver of membership probability, while minimum samples was the strongest driver of cluster size. No single configuration maximized all three criteria together, the configurations that scored best on DBCV sat at the opposite edge of the tested range from the configurations that scored best on cluster size and membership probability, confirming that these criteria trade off against each other rather than moving together.

Second round. This round narrowed in on the edge identified above, to see how far each of the two competing directions could be pushed before something broke. Testing lower neighbor counts showed that very low values (2) do not simply favor size and confidence further, they collapse the UMAP embedding itself, dropping DBCV to near zero and pushing over 60% of points into noise. Testing larger minimum cluster sizes addressed a separate concern: whether DBCV would hold up smoothly as minimum cluster size grew, or instead drop sharply once it passed some threshold, a "cliff", once small natural clusters are forced to merge. No such cliff appeared, DBCV declined gradually as minimum cluster size increased rather than falling off a threshold.

Third round. With the neighbor count collapse and the absence of a DBCV cliff both established, this round fixed UMAP's number of dimensions at 16, shown in this round to have a negligible effect on any of the three criteria, and re-tested the remaining hyperparameters over a wider range to confirm that both findings from the second round still held once the search space was extended.

The final configuration, shown in Table 3.1, sits on the Pareto front across all three criteria (DBCV 0.407, mean membership probability 0.789, mean cluster size 35.3 across 665 clusters, noise ratio 0.329). It favors the validity score over maximal cluster size: nearby configurations trade a small DBCV gain for smaller clusters (e.g. minimum cluster size of 11: DBCV 0.412, mean size 33.1) or larger clusters for a DBCV loss (e.g. minimum cluster size of 25: mean size 69.4, DBCV 0.374, noise ratio 0.401). This choice reflects that Section 3.7's trend classification treats each cluster as one coherent topic, so clusters that are internally sparse or poorly separated risk conflating distinct topics into a single misleading trend.

3.6.5 From Topic Label Clusters to Item Clusters

It is important to be precise about what is actually being clustered in the previous step. HDBSCAN operates on unique topic labels, not on items. An item, meaning a YouTube video or a Stack Exchange post, is only ever assigned to a cluster indirectly, through the topics that were extracted from it in Section 3.5. From this point on, this thesis refers to a cluster of

topic labels produced directly by HDBSCAN as a **topic label cluster**, and to the corresponding set of items built from it, described below, as an **item cluster**. The two are in a one to one correspondence, every topic label cluster has exactly one item cluster built from it, but they count different things: a topic label cluster's size is the number of topic labels it contains, while an item cluster's size is the number of items assigned to it.

Once the topic labels have been clustered, we build a corresponding item cluster for every topic label cluster that HDBSCAN discovered. An item is placed into an item cluster if at least one of the topics extracted from that item was assigned to the corresponding topic label cluster, and only that subset of the item's topics, rather than the item's full topic list, is recorded as belonging to that cluster.

A direct consequence of this design is that a single item can belong to several item clusters at once, since an item can and often does discuss more than one topic. This is expected behavior rather than a duplication error, and it also means that if the number of items assigned to each item cluster is summed across all item clusters, the result will be larger than the number of unique items in the corpus, because items that discuss several topics are counted once per item cluster they contribute to. Figure 3.4 traces this for a concrete example.

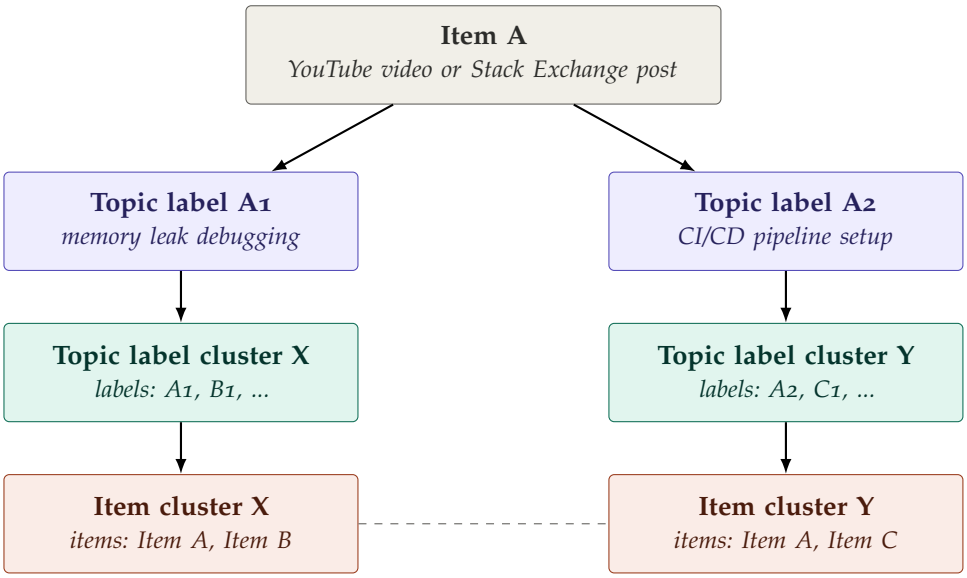


Figure 3.4: An item's topic labels can land in different topic label clusters, producing multiple item cluster assignments for the same item.

We refer to each such item-to-cluster membership, one item counted toward one item cluster, as an **item cluster assignment**. This is the unit counted whenever this thesis reports a number of assignments rather than a number of items, and it is the reason the two counts differ, as already noted above. For the remainder of this thesis, a **topic**, in the sense used by **RQ1** and **RQ2**, refers to an item cluster. The topic label cluster is the intermediate construct HDBSCAN produces on the way to it.

Table 3.2: Terminology used for the clustering stage and beyond

Term	Meaning
Topic label	A single self-contained sentence extracted from one item by the LLM, describing one Software Engineering topic it discusses.
Topic label cluster	A group of semantically similar topic labels, found directly by HDBSCAN. Its size is a count of labels.
Item	A single YouTube video or Stack Exchange post.
Item cluster	The set of items that contributed at least one topic label to a given topic label cluster. In one to one correspondence with topic label clusters, but its size is a count of items, not labels.
Item cluster assignment	One item's membership in one item cluster. An item discussing several topics contributes several assignments.
Topic (as in RQ1, RQ2)	Operationalized in this thesis as an item cluster.

Listing 3.1: Prompt used for Stack Exchange sources

```
You are an expert Software Engineering researcher specialising in identifying
    technical topics discussed by practitioners.

Your task is to analyse the provided Stack Exchange question, its tags, and its
    accepted answer (and any additional answers), and extract the distinct Software
    Engineering topics that are meaningfully addressed, following the steps below.

Step 1 - List candidates: Read the question, its tags, and all provided answers, and
    list every Software Engineering topic you notice being discussed. Use the tags as
    context clues to help you understand the domain, but do not extract a topic from a
     tag alone if it is not actually discussed in the question or answers.

Step 2 - Merge duplicates: Some of the topics from Step 1 may refer to the same
    underlying idea, mentioned once in the question and again in an answer, or phrased
     differently across answers. Merge these into a single, more general topic. For
    example, "Debugging a NullPointerException in Java" and "Handling null values
    safely in Java" describe the same idea and should become ONE topic, not two.

Step 3 - Select the final topics: From the merged list, keep between 1 and3
topics, prioritising the ones most central to the question. Do not force 3 topics if
    fewer than 3 distinct ideas are truly present.

Formatting rules for each final topic:
- Each topic must be a single, self-contained sentence, specific enough for a
  reader to understand its full scope without further context.
  - Good: "Using AST-based incremental linting in IDEs to avoid re-parsing
    unmodified code"
  - Bad: "IDE linting" (too vague, not self-contained)
- Only include topics relevant to Software Engineering practice (tools,
  techniques, processes, architectures, languages, frameworks, methodologies).

Output format:
Return ONLY a single JSON object with exactly two fields, nothing else
(no prose, no markdown fences, no explanation):
- "candidate_topics": the merged list from Step 2, as an array of strings.
- "topics": the final array of 1-3 topic-label strings from Step 3.

Example: {"candidate_topics": ["Topic A", "Topic B (same idea as A)", "Topic C"],
"topics": ["Topic A", "Topic C"]}
```

Listing 3.2: Prompt used for YouTube sources

```
You are an expert Software Engineering researcher specialising in identifying
technical topics discussed by practitioners.

Your task is to analyse the provided YouTube video content (title, description,
and transcript) and extract the distinct Software Engineering topics that are
meaningfully discussed, following the steps below.

Step 1 - List candidates: Read through the whole video content and list every
Software Engineering topic you notice, even if some of them end up describing
the same underlying idea in different words.

Step 2 - Merge duplicates: Some of the topics from Step 1 may refer to the same
underlying idea, just mentioned in different parts of the video or phrased \
differently. Merge these into a single, more general topic. For example, \
"Setting Up a CI/CD Pipeline with GitHub Actions" and "Automating Deployments
Using GitHub Actions" describe the same idea and should become ONE topic, not two.

Step 3 - Select the final topics: From the merged list, keep between 1 and 8
topics, prioritising the ones most central to the video. Do not force 8 topics
if fewer than 8 distinct ideas are truly present.

Formatting rules for each final topic:
- Each topic must be a single, self-contained sentence, specific enough for a
  reader to understand its full scope without further context.
  - Good: "Deploying a Flask Application to AWS using Docker and ECS"
  - Bad: "Flask" or "deployment" (too vague, not self-contained)
- Only include topics relevant to Software Engineering practice (tools,
  techniques, processes, architectures, languages, frameworks, methodologies).
- Ignore off-topic tangents, sponsor messages, or unrelated chatter.

Output format:
Return ONLY a single JSON object with exactly one field, nothing else
(no prose, no markdown fences, no explanation):
- "topics": the final array of 1-8 topic-label strings from Step 3.

Example: {"topics": ["Topic A", "Topic C"]}
```

3.6.6 Naming Clusters

At this point we have topic label clusters produced by HDBSCAN, together with the item cluster built from each of them as described in Section 3.6.5. Since the two are in one to one correspondence, we assign each topic label cluster a single name that also identifies its corresponding item cluster throughout the rest of this thesis.

The name, used throughout the rest of the thesis to reference a cluster, is a TF-IDF based name. We fit TF-IDF once across all topic labels in the dataset, and for each topic label cluster we take the top 5 terms according to this global fit and join them together to form the cluster's label, for example "Ai · Software Development · Software · Development · Using Ai". Because this name is composed from several terms, it is a composite description of the cluster rather than a label any single item actually used.

3.6.7 Noise Handling

As mentioned in Section 3.6.4, HDBSCAN does not force every point into a cluster. Points that do not sit in a sufficiently dense region are instead assigned to a special noise cluster. We treat a noise label as meaning that the topic in question does not represent a coherent, recurring software engineering theme, and for this reason we exclude every topic label that was classified as noise. This exclusion is carried forward into the frequency and trend analysis described in Section 3.7. An item (YouTube video or Stack Exchange post) is only left out of the analysis entirely if every one of its extracted topics falls into noise, otherwise it still contributes through whichever of its topics were assigned to a topic label cluster.

3.7 Analyzing Topic Evolution

The final stage of the pipeline analyzes how prominent each cluster produced in Section 3.6 is overall, how discussion is distributed across topics within a given year, and how the prevalence of each cluster changes across the seven years considered in this thesis (2020-2026). Unless stated otherwise, every count and share computed in this section is measured over item clusters, not topic label clusters. A cluster's prominence here reflects how many items were assigned to it, not how many topic labels HDBSCAN grouped into it, these are distinct quantities, reported separately in Section 3.6.4 and Section 4.1.2.2. As established in Section 3.6.7, topic labels that were classified as noise by HDBSCAN, and consequently the item cluster contribution they would otherwise have produced, are already excluded before this stage begins. Because a single item (a YouTube video or a Stack Exchange post) can be assigned to more than one item cluster, the total number of item cluster assignments considered below is larger than the number of unique items in the corpus, as already noted in Section 3.6.5. Figure 3.5 shows this pattern for a small example of three items.

Four independent trend detection methods are combined in this section: linear regression, polynomial regression, the Mann-Kendall trend test, and Negative Binomial regression. The general statistical background and motivation for using these four methods together is

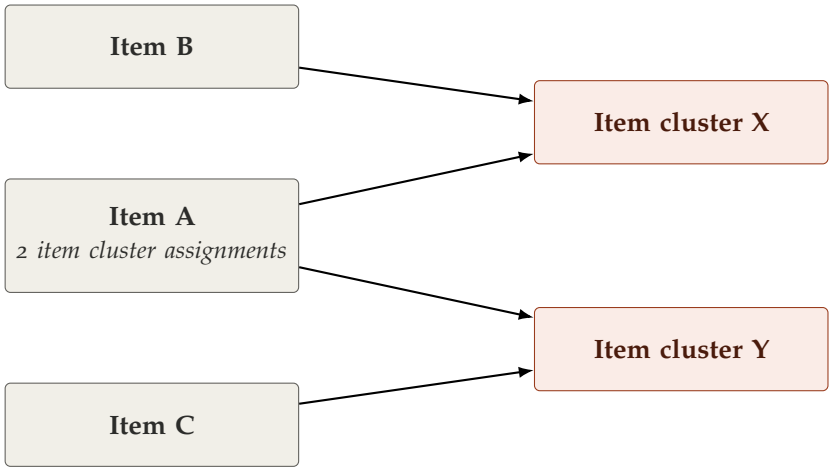


Figure 3.5: Item cluster membership is not exclusive: an item with several topics can belong to several item clusters at once.

introduced in the Background Chapter 2, the subsections below focus on the exact formulas, parameter values, and thresholds used in this thesis.

3.7.1 Cluster Size and Yearly Share

For every cluster C_j, we measure how prominent it is in two complementary ways. The first is its size, sometimes also referred to as its absolute frequency, defined as the total number of item cluster assignments belonging to that cluster across the whole seven year window:

$$f(C_j) = \sum_{i=1}^{n} a_{i,j}$$

where $a_{i,j}$ is 1 if item i is assigned to item cluster C_j and 0 otherwise, and n is the total number of items entered into clustering. Because an item can be assigned to more than one item cluster, as established in Section 3.6.5, $a_{i,j}$ is indexed by both the item and the cluster rather than picking out a single cluster per item, an item with several topics can contribute a value of 1 for several different values of j at once. Sorting item clusters by $f(C_j)$ identifies which Software Engineering topics are discussed most often overall, answering the first part of **RQ2**. We use the term size for $f(C_j)$ throughout the rest of this thesis, since it is easier to follow than absolute frequency, and reserve the latter term for the rare place where the distinction from a proportion needs to be made explicit. Unless explicitly qualified as topic label cluster size, as in Section 3.6.4 and Section 4.1.2.2, every later use of cluster size in this thesis refers to this item cluster definition of $f(C_j)$.

The second is a cluster's share for a given year Y, defined as:

$$p_j^{(Y)} = \frac{f(C_j, Y)}{\sum_k f(C_k, Y)}$$

where $f(C_j, Y)$ is the number of item cluster assignments made in year Y that were assigned to item cluster C_j and the denominator is the total number of item cluster assignments

made across all item clusters in year Y. We use the yearly share $p_j^{(Y)}$, rather than its raw yearly count, as the input to the diversity measure and the trend analysis below (Sections 3.7.4- 3.7.6), since the total amount of data collected (Section 3.4) is not necessarily the same across all seven years and using raw counts could make a cluster look like it is growing simply because more data was collected overall in that year. The one exception is Negative Binomial regression (Section 3.7.7), which is fitted directly on the raw yearly counts together with an offset that accounts for this same difference in yearly data volume, for reasons explained in that section.

3.7.2 Topic Diversity (Entropy)

To capture not just how a single cluster behaves but how discussion is distributed across all clusters in a given year, we additionally compute the Shannon entropy of the yearly share distribution:

$$H(Y) = -\sum_{j=1}^{m} p_j^{(Y)} \log_2(p_j^{(Y)})$$

where $p_j^{(Y)}$ is the share of cluster C_j in year Y as defined in Section 3.7.1 and m is the number of active cluster in year Y (clusters with at least one item assignment in that year) and the logarithm is taken base 2, so that $H(Y)$ is reported in bits. A low entropy value means that discussion in that year is concentrated on a small number of dominant clusters, while a high entropy value means that discussion is spread evenly across many different clusters.

This measure is computed exclusively on the yearly share of item cluster assignments defined in Section 3.7.1, and not on the corresponding share of unique items. This is a deliberate choice, because a single item can be assigned to more than one cluster, the unique item shares for a given year do not sum to one across clusters, and therefore do not form a valid probability distribution over which Shannon entropy is defined. The item cluster assignment share does not have this problem, since every assignment belongs to exactly one cluster by construction.

Tracking $H(Y)$ across the seven years complements the per cluster trend analysis in Sections 3.7.4 through 3.7.9, while those methods describe whether one individual cluster is becoming more or less prominent, entropy describes whether practitioner discussion as a whole is becoming more concentrated or more diverse, which is a natural part of describing how topics evolve over time for RQ2.

3.7.3 Descriptive Statistics and Ranking of Clusters by Size

For a general description of the clusters found by the pipeline, we report the total number of clusters, together with the mean, median, and standard deviation of $f(C_j)$ across all clusters. We additionally report the k clusters with the highest and lowest $f(C_j)$ (size), using $k = 10$. For every cluster in these two lists, we report both its share of all item cluster assignments and its share of all unique items, since these two proportions can differ, a cluster can be large

mainly because many items are assigned to it exclusively, or mainly because it frequently co-occurs with other clusters on the same item. Reporting both values side by side lets these two situations be told apart.

3.7.4 Linear Regression

Linear regression as described in this section is diagnostic rather than classificatory. It is not used by the trend classification procedure in Section 3.7.9, which relies exclusively on the Mann-Kendall test and the Negative Binomial growth rate. It instead supports that classification indirectly, by providing part of the fit comparison that motivates choosing Mann-Kendall and Negative Binomial regression as the primary trend detection methods, and by supplying a reference curve for the trajectory figures.

For each cluster's yearly series of shares $p_j^{(Y)}$, we fit an ordinary least-squares linear regression against the year.

The resulting slope gives the rate of change per year assumed by the best fitting straight line expressed in percentage points of share per year. A positive slope indicates growing attention, a negative slope indicates declining attention, and a slope close to zero indicates a stable topic. We also record the R^2 value of the fit, to indicate how well a straight line actually describes the data and the two sided p-value for the null hypothesis that the slope is zero.

3.7.5 Polynomial Regression (Degree 2)

Like linear regression in Section 3.7.4, polynomial regression here is diagnostic rather than classificatory and is not used by the trend classification procedure in Section 3.7.9. It complements the linear fit above by supplying the second half of the fit comparison, together with its own reference curve for the trajectory figures.

We fit a second degree polynomial (a parabola) to the same yearly series of shares, using mean centered years to keep the fitted coefficients numerically stable. The sign of the leading coefficient indicates whether the trend is concave up, concave down, or effectively linear, and the resulting R^2 value is compared against the linear fit's R^2 to assess whether curvature meaningfully improves the description of the data. The exact centering procedure, coefficient reporting convention, and concavity thresholds are given in Section A.2.

3.7.6 Mann-Kendall Trend Test

We additionally run the Mann-Kendall test on each cluster's yearly series of shares. As introduced in Chapter 2, this test does not assume that the trend follows any particular shape. It instead compares every pair of yearly values and accumulates these comparisons into a single test statistic S, positive when later years tend to exceed earlier ones more often than the reverse, negative when the opposite pattern dominates, and closer to zero when increases and decreases are roughly balanced. From S we compute Kendall's τ, a

normalized version of the same statistic bounded in $[-1, 1]$, together with a p-value against a significance level of $\alpha = 0.05$.

A cluster's series is only categorized as **increasing** or **decreasing** once this p-value falls below α, taking the sign of S to set the direction. If the p-value does not reach this threshold, the series is categorized as showing **no consistent monotonic trend**, regardless of which way S happened to point. This means the categorical direction reported by this test is never separate from its significance, a cluster is only ever called increasing or decreasing here once that direction has already cleared the significance threshold, a point that matters in Section 3.7.9, where this categorical direction is used directly to assign a cluster to the Emerging or Declining category.

Because each series contains only seven yearly values, we treat its result with some caution and report the p-value alongside the categorical trend direction, rather than only a significant or not significant label, since the test has limited statistical power with this few observations.

3.7.7 Negative Binomial Growth Rate

As a primary estimate of each cluster's annual growth rate, we fit a Negative Binomial regression to the cluster's raw yearly item counts, rather than to its share. We choose count regression for this purpose because a cluster's yearly count is a genuine, non-negative count that can legitimately be zero in a given year, and continuous methods such as the two regressions above are not built to handle such zero counts in a growth rate estimate, an endpoint percentage change is undefined when the first year is zero, a compounded annual growth rate breaks for the same reason, and averaging yearly percentage changes is biased upward, since growth compounds multiplicatively rather than additively. Negative Binomial regression treats a zero count as an ordinary, expected observation rather than a special case, which makes it suited for this data.

The fitted model is:

$$Y_t \sim \mathrm{NB}(\mu_t, \alpha)$$
$$\log(\mu_t) = \beta_0 + \beta_1 \cdot t + \log(population_t)$$

where Y_t is the observed number of item cluster assignments to the cluster in year t, modeled as a random variable following a Negative Binomial distribution rather than a fixed quantity. μ_t is the expected count in year t under the model, that is, the mean of this distribution, and it is μ_t, not the observed count Y_t itself, that is linked to the linear predictor through the logarithm. This distinction matters precisely because μ_t is always strictly positive by construction, so the model never requires taking the logarithm of an observed zero count. β_0 is the intercept, representing the baseline expected log rate of the series. β_1 is the slope, representing the estimated change in the log rate per year, and is the coefficient of interest for estimating the cluster's growth rate. $population_t$ is the total number of item cluster assignments across all clusters in year t, and $\log(population_t)$ is included as a fixed offset rather than an estimated term. This offset accounts for the same difference in yearly data volume that motivates using share instead of raw counts elsewhere in this section, while

still allowing the model to operate on counts and therefore to handle zero years correctly. Finally, α is the dispersion parameter of the Negative Binomial distribution, governing the variance of Y_t around μ_t, and is discussed further below.

The dispersion parameter of the Negative Binomial distribution, α, is estimated from the data by maximum likelihood, rather than fixed to an arbitrary constant. This choice matters in practice: fixing α to a value that is too low for a given series makes a genuinely noisy series appear falsely significant, while fixing it too high makes a genuinely strong trend appear falsely non-significant. Estimating α per series avoids both failure modes.

The estimated annual growth rate is reported as:

$$growthrate(C_j) = (e^{\beta_1} - 1) \cdot 100$$

together with a p-value for the null hypothesis that $\beta_1 = 0$, testing whether the observed trend in counts is distinguishable from no systematic change over time.

Each cluster is additionally assigned one of four status labels, **Significant**, **Not Significant**, **Insufficient Data**, or **Fit Failed**. A cluster is marked **Insufficient Data**, and no fit is attempted, if fewer than three of its seven yearly counts are non-zero, this is a minimum floor on the amount of information available to the model, not a guarantee that a fit above this floor is reliable, so the reported p-value is still the deciding factor for whether a growth rate should be treated as statistically significant.

3.7.8 Secondary Growth Cross-Checks

Alongside the Negative Binomial growth rate, we compute two additional descriptive statistics per cluster, used as secondary cross-checks rather than as primary evidence of a trend.

Coefficient of variation. We compute the coefficient of variation of a cluster's yearly shares, $std(p^{(Y)})/mean(p^{(Y)}) \cdot 100$, an order independent measure of how far a series typically sits from its own average across the whole window.

Year over year volatility. We separately compute the standard deviation of the six year over year percentage changes in a cluster's yearly share, an order dependent measure of how much the series jumps from one year to the next, regardless of overall direction.

These two volatility measures answer different questions and can diverge substantially on the same series: a steadily trending cluster can have a moderate coefficient of variation but low year over year volatility, since each step is a similarly sized change in the same direction, while a cluster that is flat on average but jumps unpredictably from year to year can show high year over year volatility despite a coefficient of variation similar to a genuinely trending cluster. Neither measure has a universal threshold separating high from low, both are interpreted, once computed across the whole dataset, against their own empirical percentiles, as described in Section 3.7.10.

3.7.9 Combining the Trend Measures

For every cluster, the Mann-Kendall test (Section 3.7.6) and the Negative Binomial growth rate (Section 3.7.7) are combined into a single trend category and confidence level, following the procedure below.

Direction. The Mann-Kendall trend direction is consulted first, since it makes no assumption about the shape of the trend, if it reports an **increasing** trend, the cluster is categorized as **Emerging**, if it reports a **decreasing** trend, the cluster is categorized as **Declining**. Only when Mann-Kendall reports no monotonic trend does the classification fall back to the sign of the Negative Binomial growth rate, gated by a minimum magnitude threshold, since the Negative Binomial estimate is the primary growth rate estimate for this data: a growth rate above the threshold is categorized as **Emerging**, a growth rate below the negative of the threshold is categorized as **Declining**, and anything in between is categorized as **Stable**. A cluster for which no Negative Binomial growth rate is available at all, marked with the **Insufficient Data** status described in Section 3.7.7, is instead categorized as **Undetermined**, since a lack of usable data is a different finding from a cluster whose growth rate was estimated and found to sit within the Stable range.

Confidence. A cluster's confidence level summarizes the agreement between the two trend detection methods using their nominal, per cluster p-values, before any correction for multiple testing is applied: confidence is **High** if the Mann-Kendall p-value is below 0.05 and the Negative Binomial p-value is below 0.05, confidence is **Medium** if exactly one of the two tests is significant at this nominal level, and confidence is **Low** if neither test is significant. An **Undetermined** cluster reaches this category only when Mann-Kendall found no monotonic trend and Negative Binomial produced no p-value at all to test, so its confidence is **Low** by construction rather than by comparison against the $\alpha = 0.05$ threshold. A correction applied later in this section adjusts the threshold used when reporting counts and proportions of significant clusters overall. How that correction relates to the confidence tier defined here is addressed once the correction itself has been introduced, in Section 3.7.11.

This means that, unlike the by size ranking in Section 3.7.1, a cluster's combined category and confidence together summarize four pieces of evidence (share trajectory, Mann-Kendall direction and significance, and Negative Binomial growth rate and significance) into two labels intended to be read together, and never a category without its accompanying confidence level.

Once every cluster has been assigned a category and a confidence level, we report how many clusters fall into each combination of category and confidence. We separately report how the Negative Binomial status from Section 3.7.7 is distributed within each trend category. This lets a specific case, such as a cluster confidently labeled as **Emerging** whose Negative Binomial fit was nonetheless marked **Not Significant**, be identified and inspected individually, rather than only appearing as a count in a table.

3.7.10 Descriptive Statistics and Ranking of Clusters by Trend Metrics

In addition to the per-cluster trend classification described above, we report dataset-wide summaries of the numeric trend metrics introduced in this section: Kendall's τ and its p-value, the Negative Binomial growth rate and its p-value, and the coefficient of variation and year over year volatility from Section 3.7.8. For each of these, we report the count, mean, standard deviation, median, mode, minimum, and maximum across all clusters, together with the 25th, 50th, and 75th percentiles. As noted in Section 3.7.8, these percentiles are what let a value be described as comparatively high or low against the rest of this thesis's own dataset, rather than against an externally imported and possibly inapplicable threshold.

For the same set of metrics, we additionally report the k clusters with the highest and the k clusters with the lowest value, again using $k = 10$. This produces a ranking by trend behavior, which is a different axis from, and should not be confused with, the ranking by cluster size already described in Section 3.7.3: both are referred to as the top and bottom k clusters, but one orders clusters by how large they are overall, and the other orders them by how they are trending.

3.7.11 Multiple Testing Correction

The Mann-Kendall test and the Negative Binomial regression are each run independently on every cluster in the dataset, several hundred times in total. At a significance level of $\alpha = 0.05$, an individual test has a five percent chance of reporting a significant trend purely by chance, even when no real trend is present. Run independently across several hundred clusters, this means a non-trivial number of clusters are expected to be flagged as significant by chance alone, before any correction is applied.

To control for this, we apply the Benjamini-Hochberg procedure for false discovery rate control to the collection of p-values produced by the Mann-Kendall test, and separately to the collection of p-values produced by the Negative Binomial regression, prior to reporting any count or proportion of clusters with a statistically significant trend. This procedure adjusts the significance threshold applied to each individual p-value based on its rank within the full set of p-values being tested, controlling the expected proportion of false positives among the clusters reported as significant, rather than the probability of any single false positive as the uncorrected significance level does. Any count of significant clusters, or any proportion derived from such a count, that summarizes the Mann-Kendall test or the Negative Binomial regression on its own reflects this correction rather than the uncorrected per cluster p-values described in Sections 3.7.6 and 3.7.7. The one exception is the confidence tier defined in Section 3.7.9, which by design uses the nominal, uncorrected p-value from each test rather than the corrected one. This tier is intended as a descriptive measure of how strongly the two methods agree at face value, not as a claim that both tests remain significant after correction, and any count or proportion of clusters reported by confidence tier elsewhere in this thesis carries that same qualification.

4

Evaluation

This chapter presents the evaluation of the topic extraction and analysis pipeline described in Chapter 3. The evaluation is organized around the two research questions introduced in Section 3.2. Section 4.1 reports the results themselves, first the evidence for RQ1, whether the pipeline can reliably extract and cluster meaningful Software Engineering topics from informal text, and then the evidence for RQ2, how the frequency of these topics changes over the collected time period. Section 4.2 discusses what these results actually mean for each research question, and Section 4.3 closes the chapter by discussing the threats to validity.

4.1 Results

This section reports what the data shows. Each subsection states counts, directions and significance levels without yet judging whether a given result is strong, expected or convincing. That judgment is left for Section 4.2, where the results are connected back to the two research questions.

4.1.1 Corpus Overview

In total, 42,671 items were collected across YouTube and Stack Exchange before any pipeline processing was applied. Of these, 39,588 items were successfully carried through the extraction pipeline and entered into clustering, the remaining 3,083 items (7.2%) were discarded due to LLM call failures or zero topic returns, as detailed in Section 4.1.2.1.

Clustering these 39,588 items produced 101,679 item cluster assignments, a number that does not simply equal the item count entered into clustering, for two separate reasons. First, a single item can be assigned to more than one cluster, as described in Sections 3.6 and 3.7, which pushes the assignment count above the item count. Second, not every item entered into clustering ends up getting assigned to a cluster at all, HDBSCAN marks a portion of items as noise rather than forcing them into a cluster, which pushes the assignment count below the item count. In this corpus, 14.1% of items entered into clustering were left unassigned as noise. Both effects are present at once, so the gap between item count and assignment count reported here should not be read as evidence of either effect on its own, the full breakdown of how well the clustering covers the corpus is given as the item level coverage ratio in Section 4.1.2.2.

The corpus combines two platforms, YouTube and Stack Exchange. Of the 39,588 items entered into clustering, 18,568 (46.9%) come from YouTube and 21,020 (53.1%) come from Stack Exchange. The Stack Exchange portion is itself split across two sites, softwareengineering and stackoverflow, with 16,667 and 4,353 items respectively. This split is stated explicitly rather than folded into a single Stack Exchange total, since one site contributing a much larger share than the others would affect how far any Stack Exchange cluster can be read as representative of Stack Exchange as a whole, rather than of just one of its two sites.

Table 4.1 breaks down the non-noise items (items assigned to at least one item cluster), and item cluster assignments, year by year from 2020 to 2026. 2026 stands out as a comparatively low count year at every stage, with only 1,719 non-noise items, against 3,547 to 7,982 items for the other years. This is an indicator of how much usable data exists for this year, not a sign that the topics of software engineering itself was discussed less recently, and it should be kept in mind whenever a later trend figure includes 2026 among its data points.

Table 4.1: Non-noise items and item cluster assignments per year

Year	Non-noise items	Item cluster assignments
2020	7,982	13,886
2021	5,621	11,307
2022	4,304	11,460
2023	3,547	12,363
2024	5,258	23,868
2025	5,561	21,591
2026	1,719	7,204

4.1.2 RQ1: Pipeline Feasibility

RQ1 asks to what extent an automated pipeline can extract meaningful Software Engineering topics from informal, conversational text, before any claim is made about what those topics are or how they change over time. Answering this requires reporting on two pipeline stages separately, since each stage contributes a distinct piece of evidence toward this extent. Section 4.1.2.1 presents the results for the extraction step, the degree to which the topic labels an item receives reflect what that item discusses. Section 4.1.2.2 presents the results for the clustering step, the degree to which items that received similar topic labels end up grouped together in a way that is internally coherent.

4.1.2.1 Topic Extraction Quality

The extraction step is the first point in the pipeline where the raw transcript or post is turned into something structured. The topic labels produced at this step are the basis for every cluster and every trend reported later in this thesis. This subsection reports the results

of two checks on this step, a manual evaluation of extraction accuracy, and the failure and discard rates recorded during processing.

Manual evaluation. A stratified sample of items was manually checked per platform, YouTube and Stack Exchange, to determine whether the topics extracted for that item accurately summarize its transcript or post content. Each item received a binary judgment, accurate or not accurate. Table 4.2 reports the results.

Table 4.2: Manual extraction accuracy by platform

Platform	Items sampled	Sampling method	Judged accurate
YouTube	100	Random Sampling	82%
Stack Exchange	100	Random Sampling	94%

This evaluation was carried out by a single annotator, the author of this thesis, so no inter rater agreement figure is reported alongside these percentages.

Failure and discard rates. Not every collected item made it as far as the manual evaluation above, some were discarded earlier in the extraction step itself. Table 4.3 reports, per platform, how many items were discarded due to an outright failure of the LLM call, and how many were discarded because the model returned zero topics for that item.

Table 4.3: Extraction failure and discard rates by platform

Platform	LLM call failures	Zero topic discards
YouTube	1,557 (7.4%)	1,017 (4.8%)
Stack Exchange	0 (0%)	509 (2.4%)

Since the raw model output is retained for the zero topic discards, a manual skim of a sample of 100 such items was carried out. Of these, 0 items (0%) contained no content relevant to Software Engineering, and 100 items (100%) contained content that failed to parse into the expected output format.

Length versus failure. One open question going into this evaluation was whether longer transcripts or posts are more likely to cause an extraction failure, for instance by exceeding what the model can reliably process in one call. Table 4.4 compares the mean and median length of successfully processed items against discarded items, per platform. Length here is measured as the number of characters in the message sent to the LLM for topic extraction, not the length of the underlying video or post itself.

For YouTube, the mean length of discarded items is 102,287.09 characters against 38,211.06 characters for successful items, a ratio of about 2.7 to 1. The median length of discarded items is 48,358.5 characters against 36,169.0 characters for successful items, a ratio of about 1.3 to 1. For Stack Exchange, the gap is smaller on both statistics. Mean length is 5,543.84

Table 4.4: Item character length (LLM input message), successful versus discarded, by platform

Platform	Mean length successful	Median length successful	Mean length discarded	Median length discarded
YouTube	38,211.06	36,169.0	102,287.09	48,358.5
Stack Exchange	4,522.38	3,439.5	5,543.84	3,967.0

characters for discarded items against 45,22.38 characters for successful items, and median length is 3,967.0 characters for discarded items against 3,439.5 characters for successful items, ratios of about 1.2 to 1 and 1.15 to 1 respectively. For YouTube, the gap between the mean ratio and the median ratio is itself notable.

Topic count diagnostics. Table 4.5 reports the distribution of the number of final topics assigned per item, per platform. For each platform, it reports the mean and median number of topics per item, together with the percentage of items that fell below, at, or above the maximum allowed number of topics, eight for YouTube and three for Stack Exchange.

Table 4.5: Topics per item and cap hit rate, by platform

Platform	Cap	Mean topics per item	Median topics per item	Cap hit rate	Topics below cap	Topics above cap
YouTube	8	8.13	8.0	29.40%	42.20%	28.40%
Stack Exchange	3	1.81	2.0	12.37%	87.56%	0.70%

4.1.2.2 Clustering Quality

Section 4.1.2.1 presented the results for the extraction step. This subsection presents the results for the clustering step, whether items that received similar topic labels end up grouped together. Table 3.2 in Section 3.6.5 summarizes the terminology used throughout this subsection, topic label, topic label cluster, item, item cluster, and item cluster assignment, and is worth revisiting here since the two units, topic label clusters and item clusters, are both reported side by side below. It reports several independent signals rather than a single number, a scorecard of headline metrics, the distribution of cluster sizes, density based cluster validity, membership confidence, and a manual coherence review. How these signals relate to each other and what that implies is discussed in Section 4.2.

Scorecard. Two different units are in play in this subsection, and keeping them separate matters for reading the numbers correctly. HDBSCAN clusters individual topic label points into topic label clusters, not corpus items, so a metric such as the noise ratio describes the share of topic label points that were not assigned to any topic label cluster. Because an item can carry several topic labels, an item is only left without an item cluster assignment if

every one of its topic labels falls into the noise cluster, so item coverage is a distinct quantity from topic label noise and the two numbers are not expected to move together. Table 4.6 reports both groups of metrics separately for this reason. Each of these signals is unpacked individually below, together with its full distribution. Noise labeled points, meaning topic label points that HDBSCAN did not assign to any topic label cluster, are excluded from every metric below except the noise ratio itself, and every number in this subsection is computed on the 16 dimensional UMAP reduced embeddings, not the original sentence embedding output, since that is the space clustering was performed in. The item coverage ratio reported here is the same figure introduced in Section 4.1.1, restated here alongside the other clustering signals since it is itself a property of the clustering, not only a raw scale fact.

Table 4.6: Clustering scorecard

Topic label level	
Total topic label clusters	2470
Noise ratio	36.13%
Overall DBCV	0.3870
Mean membership probability	0.7651
Item level	
Item coverage ratio	85.9%

The noise ratio and the item coverage ratio reported above describe different quantities and are not directly comparable. 36.13% of topic label points fell into HDBSCAN noise, but this does not mean that a matching share of corpus items were discarded. Because an item can carry several topic labels, an item is only left without an item cluster assignment if none of its topic labels were assigned to a topic label cluster, so only 14.1% of items ended up with no topic label attached to any topic label cluster, the complement of the 85.9% item coverage ratio reported above. The gap between the two figures is expected rather than contradictory, and follows directly from the one to many relationship between items and topic labels described in Chapter 3.

Cluster size distribution. Across the 2,470 topic label clusters, the number of member topic labels per cluster ranges from 11 to 1,166. The mean topic label cluster size is 49.07 and the median is 32.0, with a standard deviation of 57.00. The gap between mean and median points to a right skewed distribution, most topic label clusters hold a comparatively small number of member topics while a smaller number of large clusters pull the mean upward. 71.5% of topic label clusters (1,766 of 2,470) contain fewer than 50 member topic labels, and 8.6% (213 clusters) contain 100 or more. The $90th$ percentile sits at 91 member topics and the $99th$ percentile at 314, so the long tail is concentrated in a small fraction of topic label clusters rather than spread evenly. Figure 4.1 shows this distribution directly,

making the concentration of small clusters and the long tail of large ones visible in a single plot.

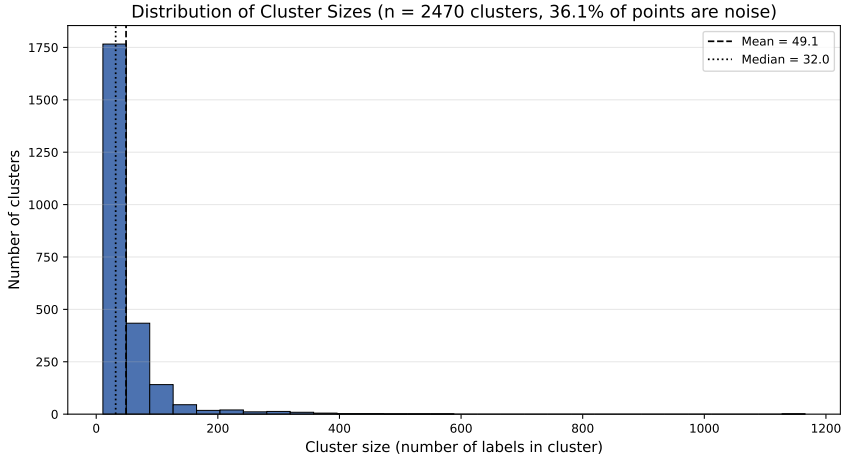


Figure 4.1: Distribution of topic label cluster sizes across all 2470 clusters

DBCV distribution. The overall DBCV score is 0.3870, on a scale from negative one to positive one, computed over the topic level clustering. Per cluster DBCV scores range from -0.9271 to 0.9999, with a standard deviation of 0.2220. The overall DBCV score is lower than the mean of the per cluster scores, 0.3870 against 0.6521, because the overall score weights each topic label point equally rather than each cluster equally, so a comparatively small number of large, weakly separated clusters pull the aggregate down even though most individual clusters score well. 43.0% of clusters score above 0.7, and only 1.3% (32 clusters) score negative, with a further 1.4% falling between 0 and 0.2. The bulk of clusters therefore show comparatively strong internal structure, and weak or negative scores are the exception rather than a common pattern. Figure 4.2 plots this distribution, showing where the small share of low and negative scoring clusters fall relative to the bulk of the data.

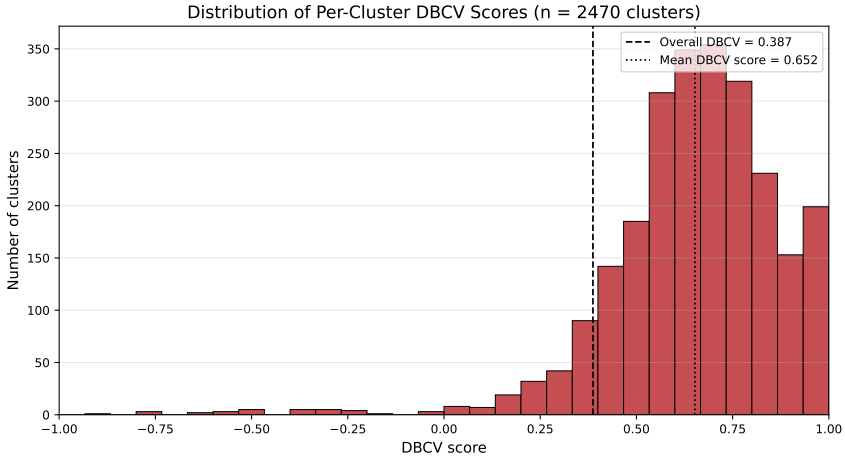


Figure 4.2: Distribution of per cluster DBCV scores

A DBCV score on its own states whether internal density structure was found, but not whether that structure differs from what an arbitrary partition of the same points would produce, since even an arbitrary partition can score above zero depending on the geometry of the embedding space. To check this, the same topic points were randomly shuffled into partitions with the same number of clusters and the same cluster size distribution as the real clustering, so that only the assignment of points to clusters changed and not the number or sizes of the clusters themselves. DBCV was computed on each of these random partitions, repeated 30 times, giving a baseline distribution of DBCV scores for what an assignment with no real density structure looks like on this data. The real clustering scored 0.3870, against a baseline mean of -0.59 with a standard deviation of 0.0007. The real score falls at the $100th$ percentile of this baseline distribution, meaning every one of the 30 random partitions scored lower than the real clustering, and the gap between the two, close to one full point on a scale from negative one to positive one, exceeds the baseline distribution's own standard deviation by several orders of magnitude.

Membership probability distribution. Mean membership probability per cluster ranges from 0.2547 to 1.0, with a median of 0.8401 and a standard deviation of 0.1515. 57.8% of clusters have a mean membership probability above 0.8, while 4.0% (98 clusters) fall below 0.5. Read alongside the DBCV distribution above, the two signals broadly agree, clusters with weak DBCV scores are also the ones most likely to show low membership confidence, rather than the two metrics disagreeing about which clusters are weak. Figure 4.3 shows the same distribution graphically, alongside the corresponding DBCV scores for each cluster.

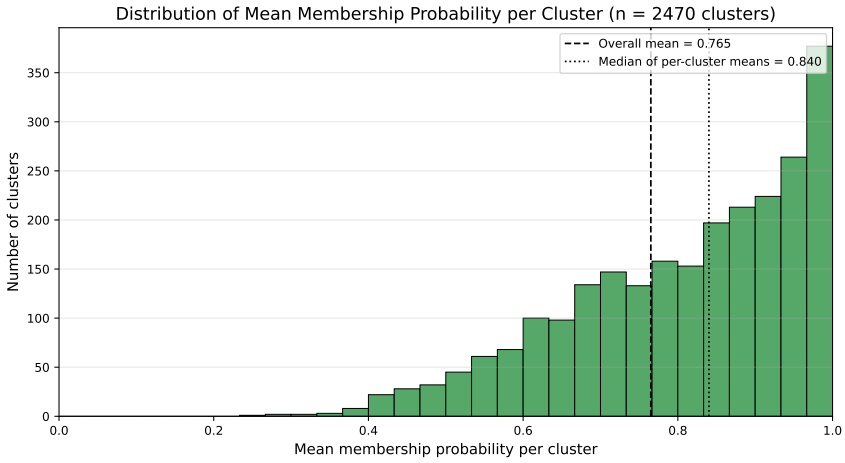


Figure 4.3: Distribution of mean membership probability per cluster

Combined health overview. Figure 4.4 presents topic level cluster size, DBCV, and membership probability together in a single view, combining the three signals reported individually above.

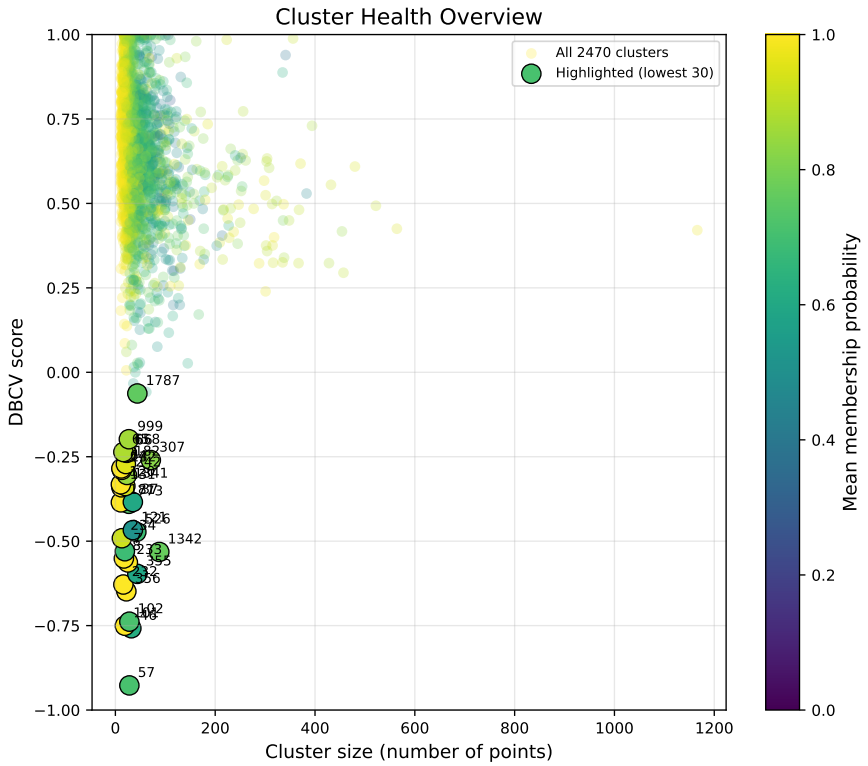


Figure 4.4: Topic level cluster size, DBCV and membership probability shown together per cluster

Manual cluster coherence review. A sample of topic label clusters was manually reviewed for whether their member topic labels share a common theme. Each sampled cluster received a binary judgment, coherent or not coherent. Table 4.7 reports the results.

Table 4.7: Manual cluster coherence review

Clusters sampled	Sampling method	Judged coherent
200	Random Sampling	89%

As with the extraction accuracy review in Section 4.1.2.1, this review was carried out by a single annotator, the author of this thesis, with no independent second rater.

4.1.3 RQ2: Topic Frequency and Evolution

RQ2 asks which Software Engineering topics appear most frequently in the corpus and how that frequency changes across the collection window, from 2020 to 2026. Answering this question reuses the same 2,470 topic label clusters, and their corresponding item clusters, already reported for RQ1 in Section 4.1.2.2. The metrics computed on them change with the question being asked, however, Section 4.1.2.2 measured topic label cluster properties (topic label counts, DBCV, membership probability) to argue that the clustering is internally sound, while this section measures item cluster properties (item counts and their change over

time) to argue how prominent and how dynamic each topic actually is. This section first reports the shape of the topic landscape across the whole collection window, then reports the results of the trend classification applied to that landscape. Section 4.1.3.1 reports the size and concentration of the item clusters across the full collection window. Section 4.1.3.2 reports the results of applying Mann-Kendall and Negative Binomial regression to each item cluster's yearly item counts, classifying each cluster as emerging, stable, declining or undetermined.

4.1.3.1 Cluster Landscape

This subsection reports the shape of the topic landscape as it stands across the whole collection window, independent of how that landscape changes over time. It reports how large item clusters are, the largest and smallest clusters by size, and how concentrated item assignments are across the topic space.

Item cluster size, measured as the number of items a cluster received, has a mean of 41.17 and a median of 29.0 across all 2,470 clusters identified in Section 4.1.2.2, ranging from 1 to 603 with a standard deviation of 41.73. This is a view of item cluster size, distinct from the topic label cluster size distribution already reported in Section 4.1.2.2, and the two are not expected to match, since a topic label cluster with few member topic labels can still receive a large number of item assignments if those topics are common ones. Figure 4.5 shows the full distribution.

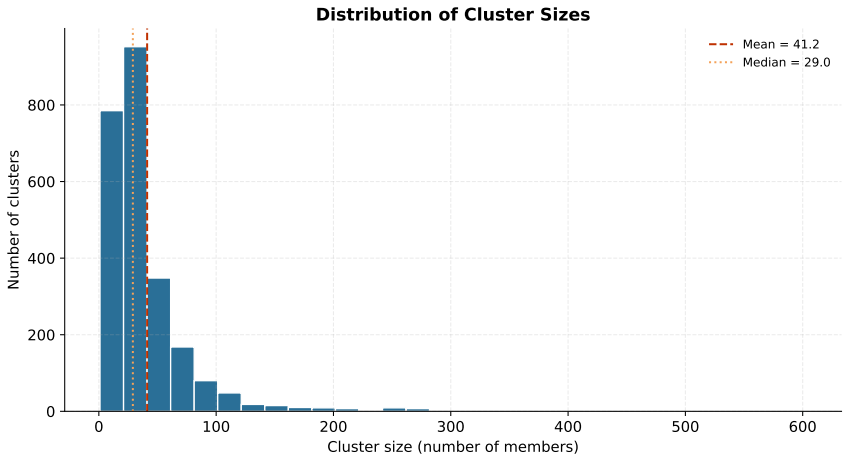


Figure 4.5: Distribution of item cluster size (total items) across all 2470 clusters

Top clusters by size. Table 4.8 reports the top 10 clusters by size, together with each cluster's share of all item cluster assignments, its share of all unique items, its trend category, and its Negative Binomial growth rate. These two shares are reported side by side rather than only one of them, since they answer different questions. A cluster can hold a large share of all assignments mainly because many items are exclusively about it, or instead because it frequently co-occurs with other topics on the same items.

Table 4.8: Top 10 item clusters by size

Rank	Cluster label	Total items	Share of assignments	Share of unique items	Trend	Confidence	Growth rate
Top 1	Kubernetes · Cluster · Kubernetes Cluster · Pod · Containerization	603	0.593%	1.77%	Declining	Medium	-12.24%
Top 2	Ai · Software · Ai Software · Software Development · Development	391	0.3845%	1.1503%	Emerging	High	153.66%
Top 3	Javascript · Framework · Javascript Framework · Javascript Fundamental · Javascript Ecosystem	375	0.3688%	1.1032%	Emerging	Low	5.92%
Top 4	Remote · Remote Work · Work · Team · Remote Team	369	0.3629%	1.0855%	Emerging	Low	8.11%
Top 5	Nextjs · Using · React · Route · Serverside	365	0.359%	1.0738%	Declining	High	-21.55%
Top 6	Setting Cicd · Pipeline Github · Cicd Pipeline · Cicd · Github Action	356	0.3501%	1.0473%	Emerging	Low	11.14%
Top 7	Code · Vscode · Visual Studio · Visual · Setting	340	0.3344%	1.0002%	Declining	High	-39.06%
Top 8	Concurrency · Thread · Concurrent · Multithreading · Lock	337	0.3314%	0.9914%	Declining	Medium	-8.87%
Top 9	Microservices · Architecture · Microservices Architecture · Microservice · Microservice Architecture	325	0.3196%	0.9561%	Declining	Medium	-15.66%
Top 10	Deployment Using · Using Github · Automating Deployment · Github Action · Automating	302	0.297%	0.88%	Emerging	Medium	15.89%

Bottom clusters by size. Table 4.9 reports the bottom 10 clusters by size, using the same columns as Table 4.8.

Table 4.9: Bottom 10 item clusters by size

Rank	Cluster label	Total items	Share of all assignments	Share of all unique items	Trend	Confidence	Growth rate
Bottom 1	Revit · Tool Revit · Wall · Type Revit · Wall Revit	1	0.0010%	0.0029%	Undetermined	Low	-
Bottom 2	Component · Ionic · Ionic Framework · Ionheader · Ionic Component	3	0.0030%	0.0088%	Declining	Low	-19.76%
Bottom 3	Gleam · Language · Safety Gleam · Language Server · Type	6	0.0059%	0.0177%	Emerging	Low	37.78%
Bottom 4	Argparse · Argument · Command-line · Python · Python Script	6	0.0059%	0.0177%	Declining	Low	-34.26%
Bottom 5	Erlang · Erlang Virtual · Scalability Erlangotp · Erlangotp · Designing Scalability	7	0.0069%	0.0206%	Declining	Low	-14.3%
Bottom 6	Vault · Secret · External Secret · Secret Operator · Hashicorp Vault	7	0.0069%	0.0206%	Declining	Low	-16.26%
Bottom 7	Tkinter · Geometry · Widget · Tkinter Controlling · Controlling Widget	8	0.0079%	0.0235%	Declining	High	-37.92%
Bottom 8	Nix · Nix Package · Reproducible · Package Manager · Package	8	0.0079%	0.0235%	Emerging	Low	46.92%
Bottom 9	Certification · Ai Certification · Ai · Professional Certificate · Certificate	8	0.0079%	0.0235%	Undetermined	Low	-
Bottom 10	Passkey · Key · Key Issue · Security Key · Creating Passkey	8	0.0079%	0.0235%	Emerging	Low	48.18%

Concentration of attention. To state how concentrated attention is across the topic space as a whole, Figure 4.6 shows the cumulative share of all item cluster assignments accounted for by clusters ranked from largest to smallest. The top 55% of clusters by size account for 80% of all item cluster assignments. This is reported alongside Table 4.8 and Table 4.9, since the top and bottom 10 tables only show the extremes, while the cumulative curve states how quickly attention concentrates across the full ranked list of clusters.

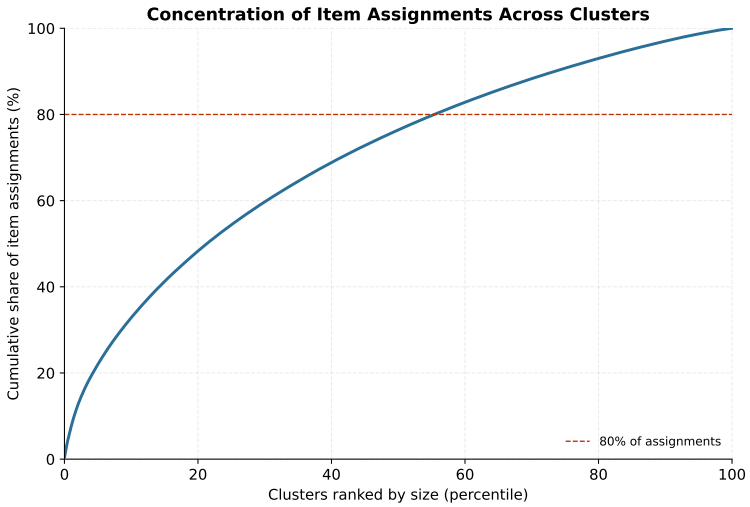


Figure 4.6: Cumulative share of item cluster assignments by cluster rank, largest to smallest

Beyond size. Clusters also differ in how many of the seven years in the collection window they were active in. Cluster lifespan, measured as the number of active years out of seven, has a mean of 6.1 and a median of 6.0, with a standard deviation of 1.1. Only 0.004% of clusters are short lived, active in only one or two years, while 48% are persistent, active across most or all of the window. Figure 4.7 shows the full distribution of active years across all clusters.

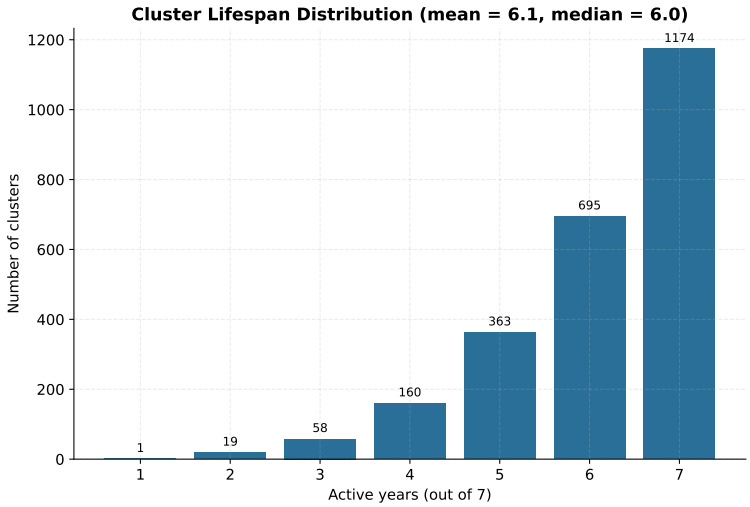


Figure 4.7: Distribution of cluster lifespan, measured as number of active years out of seven

Figure 4.8 reports the Shannon entropy of the yearly share distribution, as defined in Section 4.7.2 Section 3.7.2, for each year in the study window. The value starts at 10.096901 bits in 2020, rises through 2021, 2022 and 2023, and reaches its highest point at 10.638081 bits in 2024. It remains close to this level through 2025 before decreasing to 10.046585 bits in 2026. This figure presents the overall spread of discussion across active clusters for each year, complementing the size distribution reported above, which describes individual cluster

prominence rather than how evenly attention is divided across the full set of active clusters in a given year.

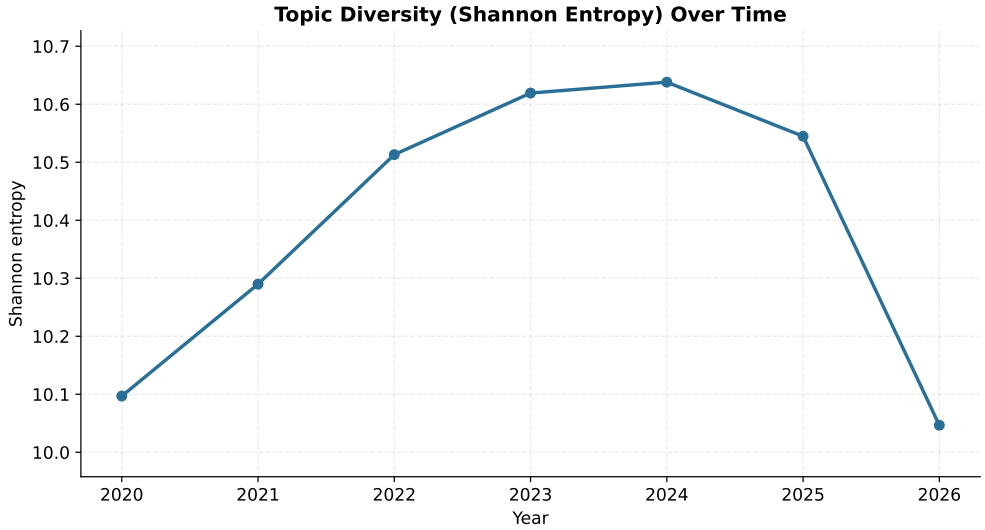


Figure 4.8: Shannon entropy of the yearly item cluster assignment share, 2020 to 2026

4.1.3.2 Trend Classification Results

Each cluster's yearly item counts are classified into one of four categories, emerging, stable, declining or undetermined, using the classification rule already introduced in Section 3.7.9. Mann-Kendall direction is used first, with Negative Binomial growth rate as a fallback when Mann-Kendall does not detect a significant monotonic trend. Clusters for which the Negative Binomial regression cannot produce a usable growth rate, typically because too few years have a non-zero item count, are placed in the undetermined category rather than folded into stable, since the two situations mean different things: a cluster genuinely holding steady across the years is a different finding from a cluster where the available data is too sparse to say either way.

Each cluster's yearly item counts are classified into four categories. Emerging accounts for 1,234 clusters, split into 185 high confidence, 246 medium confidence and 803 low confidence. Stable accounts for 393 clusters, all at low confidence. Declining accounts for 823 clusters, split into 235 high confidence, 201 medium confidence and 387 low confidence. Undetermined accounts for 20 clusters, all at low confidence. Confidence tier is assigned alongside the category itself, following the same classification rule introduced in Section 3.7.9, and is based on the nominal, uncorrected p-value from each test rather than the Benjamini-Hochberg corrected p-value introduced in Section 3.7.11. Figure 4.9 shows the category totals graphically, and Figure 4.10 shows the confidence tier breakdown for each category.

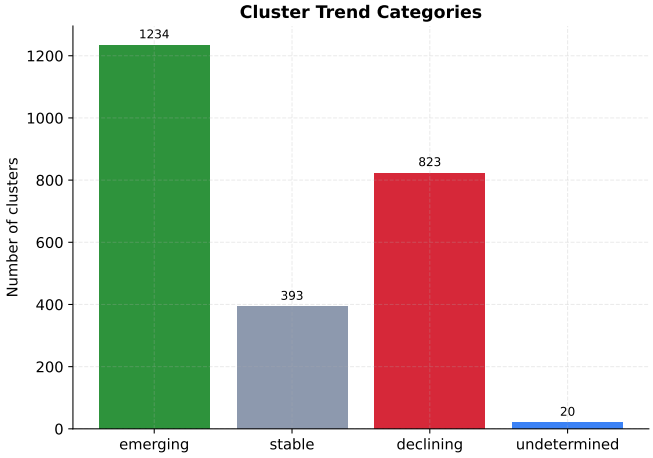


Figure 4.9: Number of clusters per trend category, emerging, stable, declining and undetermined

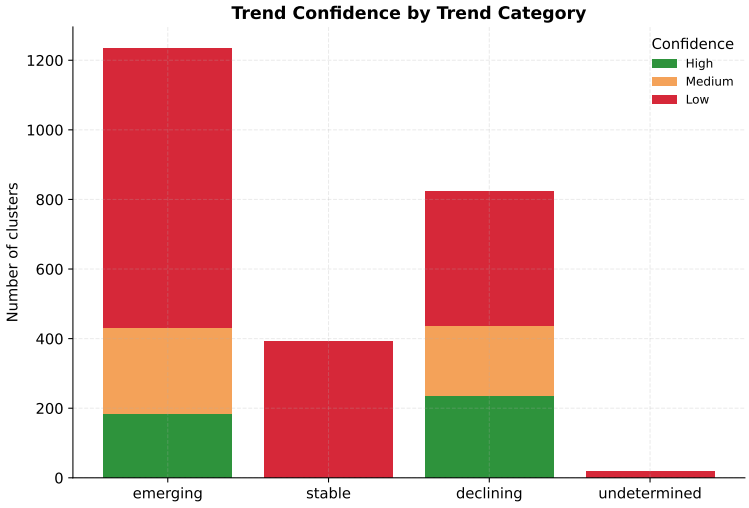


Figure 4.10: Trend category counts broken down by confidence tier, high, medium and low

Growth rate distribution. Negative Binomial growth rate, expressed as a percentage change in item count per year, has a mean of 9.5 and a median of 5.2 across all clusters for which a growth rate could be computed, with a standard deviation of 46.4. Figure 4.11 shows the full distribution.

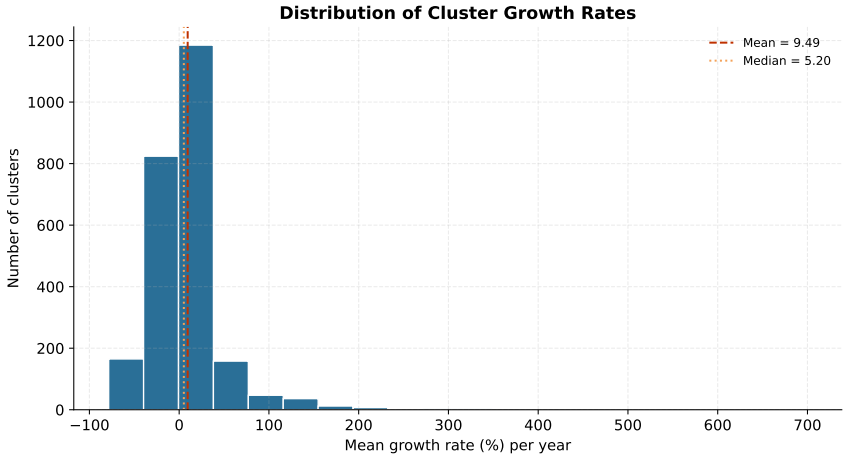


Figure 4.11: Distribution of Negative Binomial growth rate across all clusters with a usable fit

Highest and lowest growth clusters. Table 4.10 reports the top 10 clusters by growth rate, together with each cluster's size, its Negative Binomial status, and its trend category. Table 4.11 reports the bottom 10 clusters by growth rate, using the same columns.

Table 4.10: Top 10 clusters by Negative Binomial growth rate

Rank	Cluster label	Cluster size	Growth rate	NB status	Trend category	Confidence
Top 1	Coding Agent · Coding · Agent · Agentic Coding · Ai Coding	35	699.35%	Significant	Emerging	High
Top 2	Vibe · Vibe Coding · Coding · Context Vibe · Rise Vibe	36	490.06%	Significant	Emerging	High
Top 3	Claude · Claude Code · Code · Feature · Clibased	29	449.36%	Significant	Emerging	Medium
Top 4	Agent · Identity · Authorization Agent · Agent Identity · Authorization	19	399.18%	Significant	Emerging	Medium
Top 5	Aigenerated Code · Aigenerated · Code · Security · Code Security	22	379.77%	Significant	Emerging	High
Top 6	Context Engineering · Context · Engineering · Contextdriven · Contextdriven Agility	21	355.58%	Significant	Emerging	High
Top 7	Specdriven · Specdriven Development · Development · Spec · Development Ai	31	347.55%	Significant	Emerging	Medium
Top 8	Construction · Agile Construction · Agile · Lean · Lean Construction	15	331.53%	Significant	Emerging	High
Top 9	Mcp · Model Context · Context Protocol · Protocol · Protocol Mcp	65	319.54%	Significant	Emerging	Medium
Top 10	Interview · Ai · Ai Technical · Ai Interview · Technical Interview	20	294.79%	Significant	Emerging	Medium

Table 4.11: Bottom 10 clusters by Negative Binomial growth rate

Rank	Cluster label	Cluster size	Growth rate	NB status	Trend category	Confidence
Bottom 1	View · Binding · View Binding · Fragment · Android	29	-78.52%	Significant	Declining	High
Bottom 2	Flutter · Button · Color · Customizing · Elevatedbutton	51	-74.66%	Significant	Declining	Medium
Bottom 3	Blob · Storage · Azure · Azure Blob · Blob Storage	16	-74.33%	Significant	Declining	Medium
Bottom 4	Time Complexity · Complexity Analysis · Complexity · Time · Analysis	28	-72.22%	Significant	Declining	Medium
Bottom 5	Autocomplete · Material Ui · Ui Autocomplete · Material · Ui	23	-68.34%	Significant	Declining	High
Bottom 6	Multipartformdata · Multipartformdata Request · Formdata · Request · Using Formdata	16	-68.22%	Significant	Declining	High
Bottom 7	Testing · React Testing · Testing Library · React · Element	18	-67.92%	Significant	Declining	Medium
Bottom 8	App Action · App · Action · Android · Testing App	16	-67.88%	Not Significant	Declining	Low
Bottom 9	Navigation Component · Navigation · Viewpager2 · Android Navigation · Android	16	-65.6%	Significant	Declining	High
Bottom 10	Jinja2 · Html · Flask · Using Jinja2 · Template	10	-65.6%	Significant	Declining	Medium

Mann-Kendall view. Independent of the Negative Binomial fit, the Mann-Kendall test finds a statistically significant monotonic trend, increasing or decreasing, for 451 of the 2,470 clusters. Of these, 200 clusters (8.1% of all clusters) show an increasing trend and 251 clusters (10.2%) show a decreasing trend. The remaining 2,019 clusters (81.7%) show no significant trend. Figure 4.12 shows the distribution of Mann-Kendall tau statistics across all clusters, and Figure 4.13 shows the direction counts graphically.

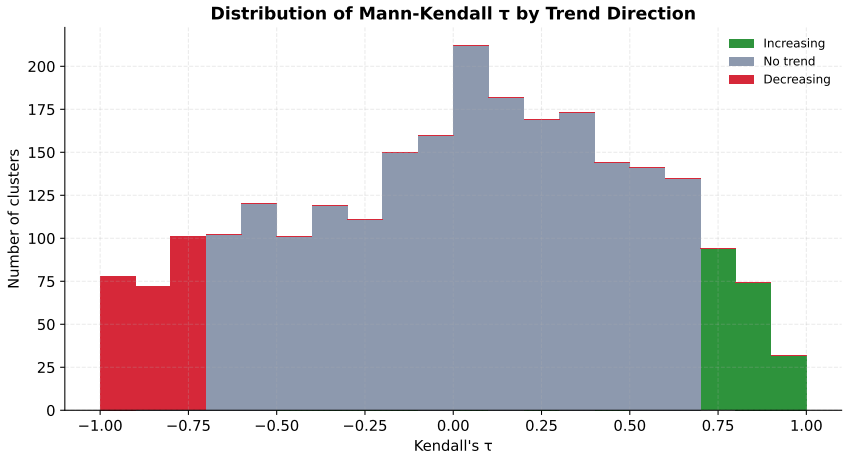


Figure 4.12: Distribution of Mann-Kendall tau statistic across all clusters

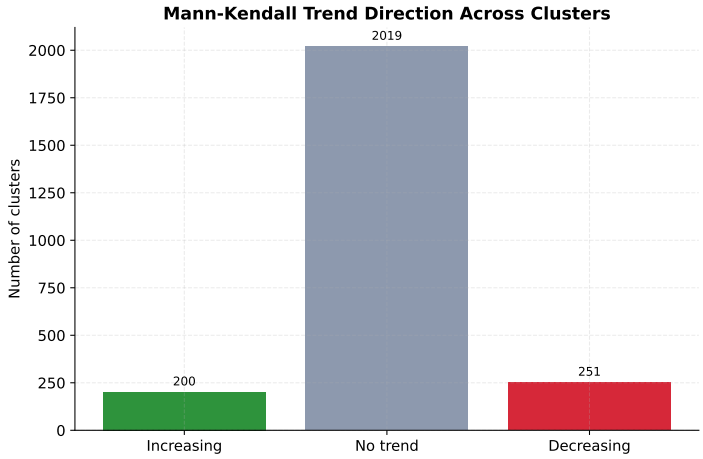


Figure 4.13: Number of clusters per Mann-Kendall trend direction, increasing, decreasing and no trend detected

Method agreement. Of the 1,234 emerging clusters, 416 are also significant under Negative Binomial regression and 818 are not. All 393 stable clusters are not significant under the Negative Binomial regression. Of the 823 declining clusters, 420 are also significant under Negative Binomial regression and 403 are not. All 20 undetermined clusters have insufficient data for Negative Binomial regression. Figure 4.14 shows this cross tabulation as a heatmap, stating how often the two agree and how often they diverge, as an explicit, named check rather than something left implicit in the category counts above.

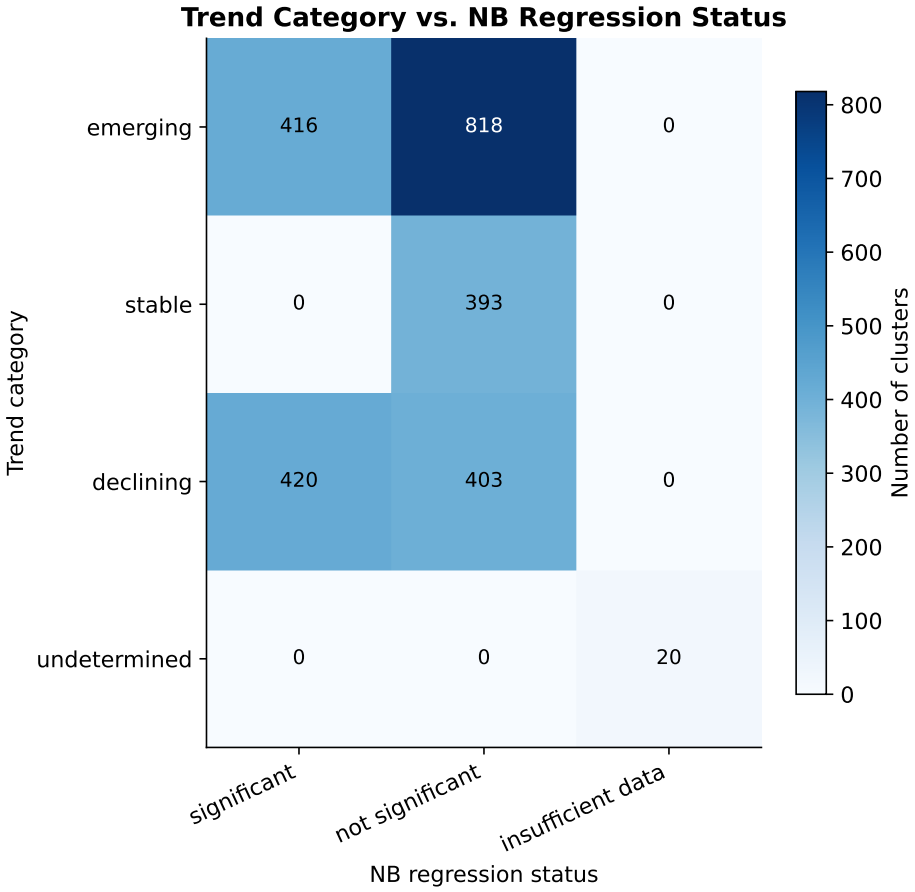


Figure 4.14: Trend category against Negative Binomial regression status, shown as a heatmap

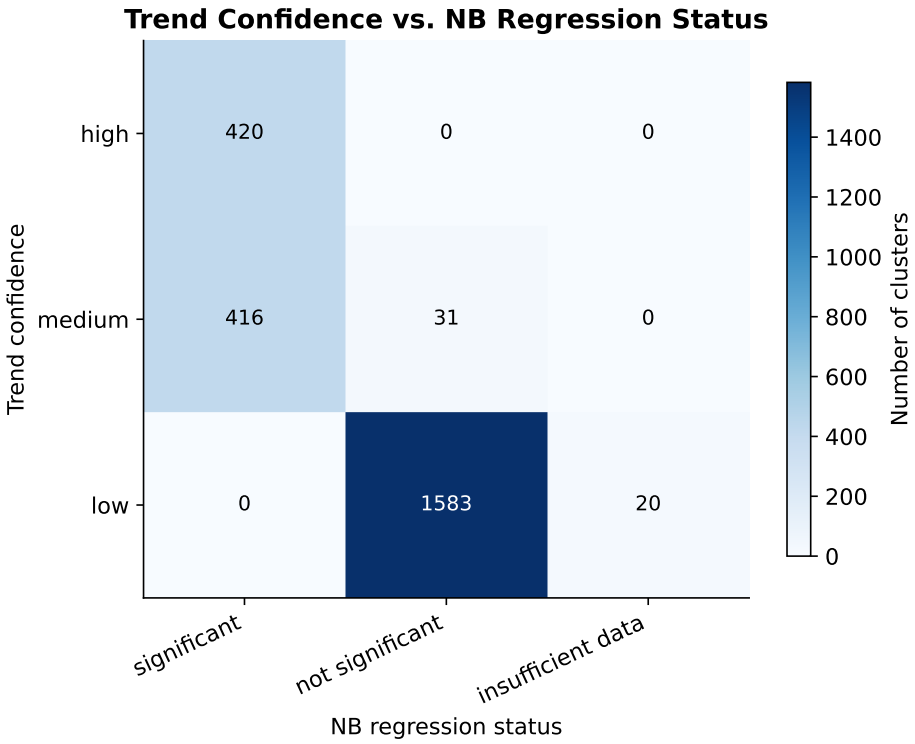


Figure 4.15: Trend confidence tier against Negative Binomial regression status, shown as a heatmap

Volatility. Coefficient of variation and year over year volatility of item counts are reported together across all clusters, since a cluster can score high on one and low on the other, and neither alone distinguishes a cluster that is smoothly trending from one that moves without a clear direction from year to year. Coefficient of variation has a mean of 80.06%, a median of 73.58%, and a standard deviation of 35.65%. Year over year volatility has a mean of 124.96%, a median of 95.5%, and a standard deviation of 121.0%. Figure 4.16 and Figure 4.17 show the full distributions separately, including the median for each, and Figure 4.18 plots the two measures against each other per cluster to show whether they agree.

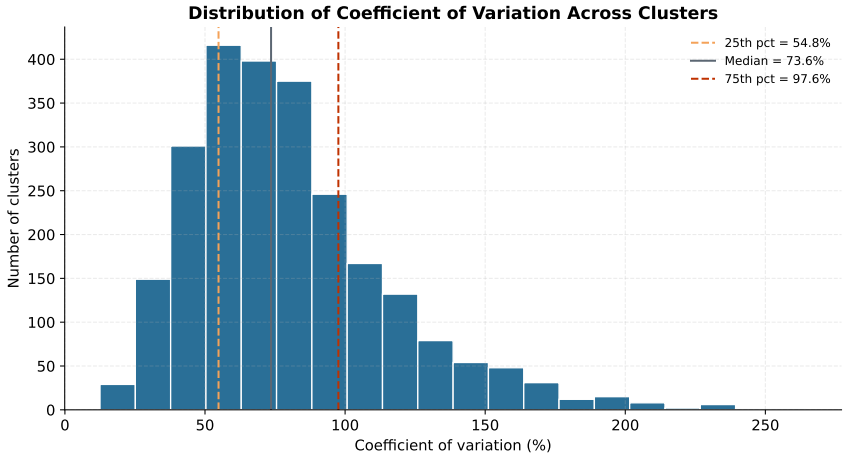


Figure 4.16: Distribution of coefficient of variation across all clusters

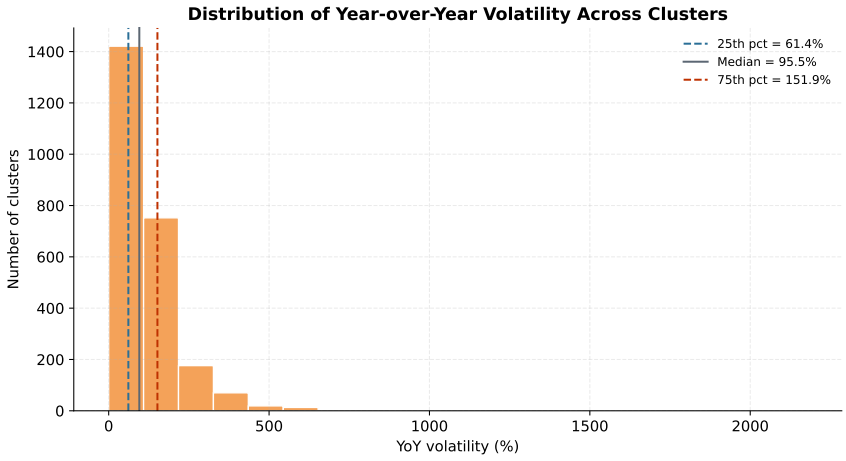


Figure 4.17: Distribution of year over year volatility across all clusters

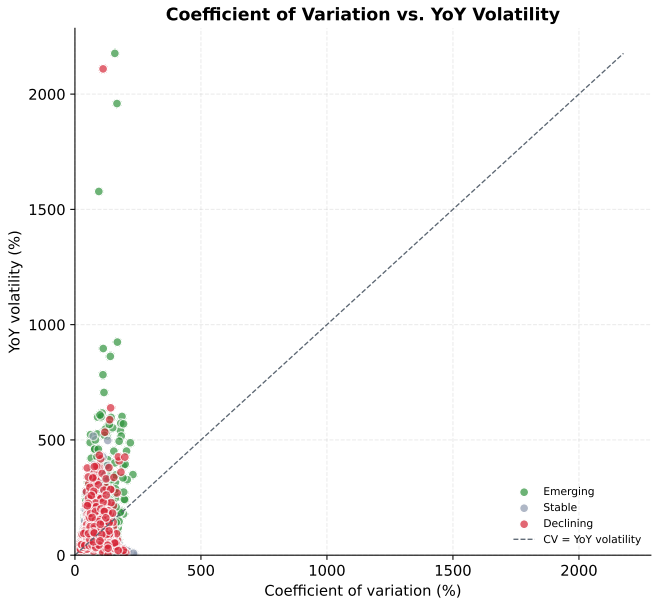


Figure 4.18: Coefficient of variation plotted against year over year volatility, per cluster

Figure 4.19 shows the yearly trajectory of the 6 clusters with the highest coefficient of variation, and Figure 4.20 shows the 6 with the lowest. Figure 4.21 and Figure 4.20 show the same, ranked by year over year volatility instead.

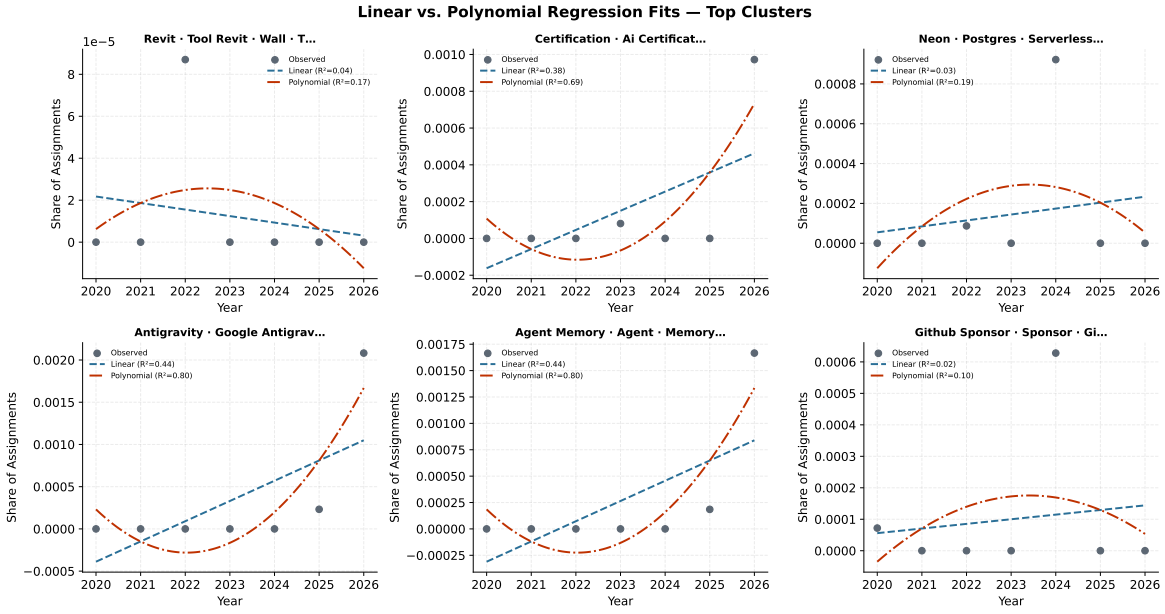


Figure 4.19: Yearly item counts for the 6 clusters with the highest coefficient of variation

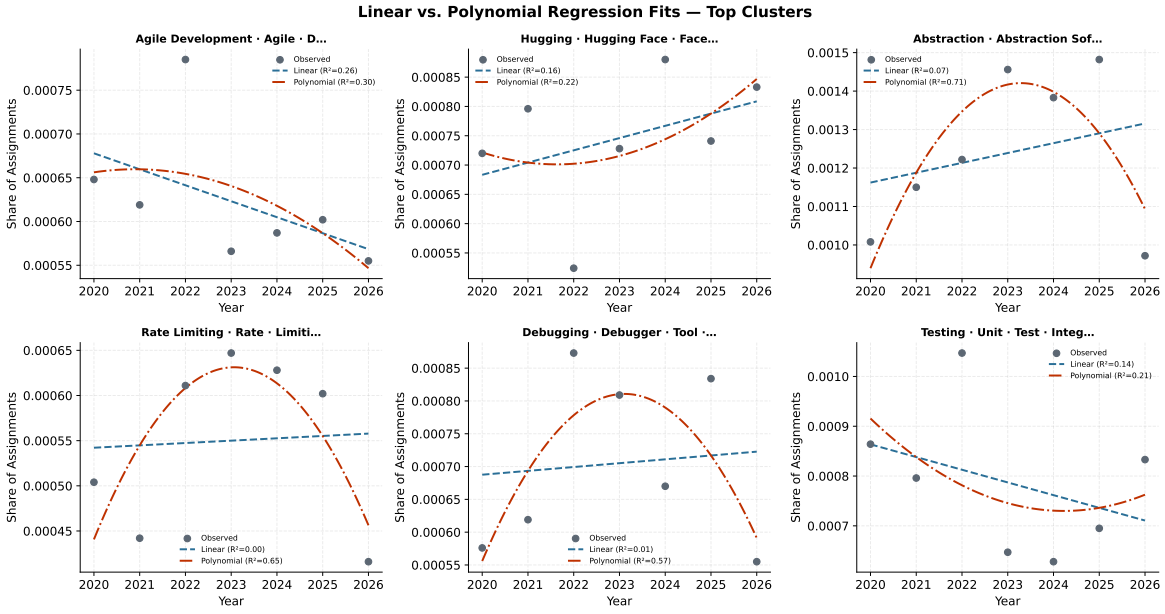


Figure 4.20: Yearly item counts for the 6 clusters with the lowest coefficient of variation

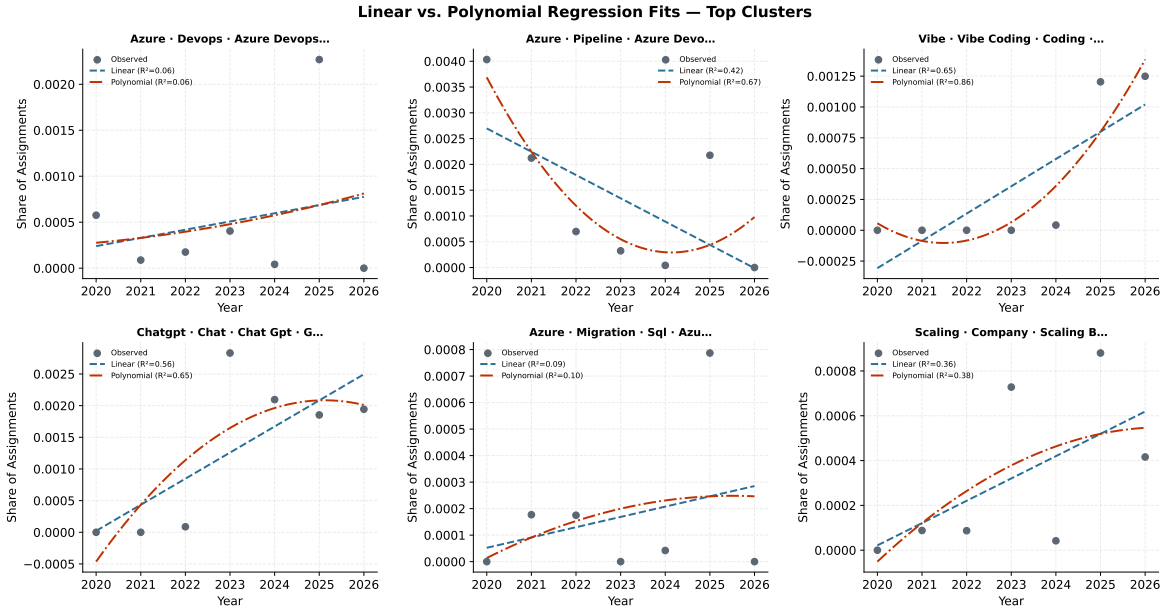


Figure 4.21: Yearly item counts for the 6 clusters with the highest year over year volatility

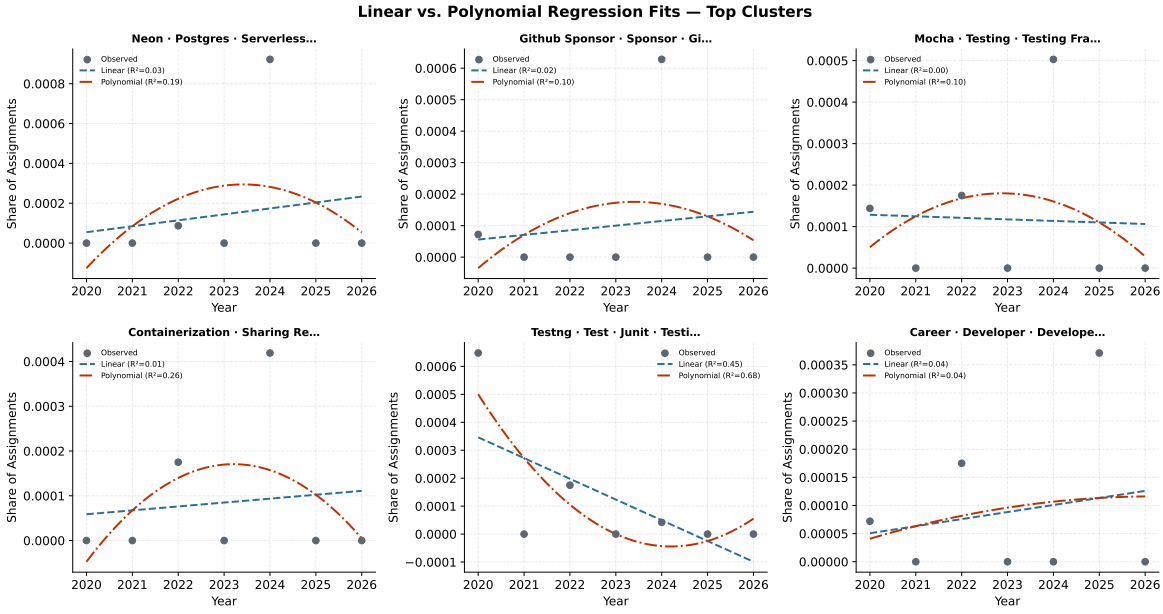


Figure 4.22: Yearly item counts for the 6 clusters with the lowest year over year volatility

Multiple testing correction. With thousands of clusters tested three times each, some raw significant results are expected by chance alone. Under the raw 0.05 cutoff, 660 of the 2,470 clusters tested with linear regression are flagged significant, 451 of the 2,470 clusters tested with Mann-Kendall are flagged significant, and 836 of the 2,450 clusters tested with Negative Binomial regression are flagged significant. After Benjamini-Hochberg correction at $\alpha = 0.05$, this drops to 0 for linear regression, 0 for Mann-Kendall, and 581 for Negative Binomial regression. Figure 4.23 shows these before and after counts graphically, per test.

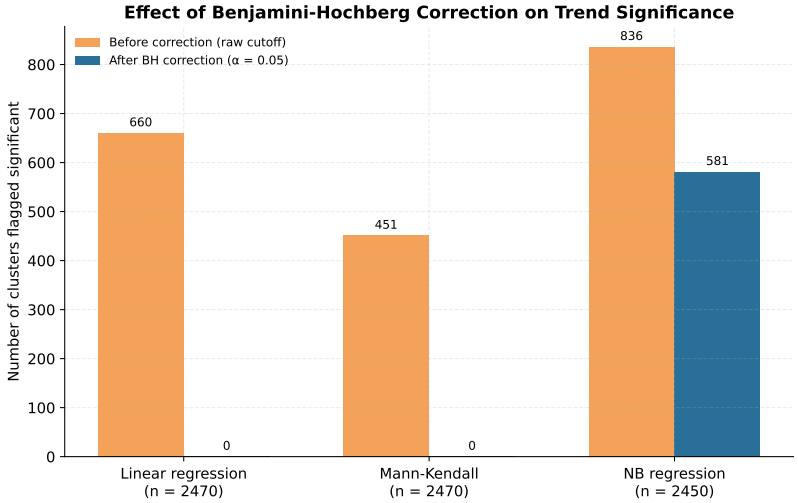


Figure 4.23: Number of significant clusters per test before and after Benjamini-Hochberg correction

Case studies. To make the categories above concrete, this section presents two small named sets of individual clusters. Table 4.12 shows three clusters chosen for the strength of their trend signal, the two highest growth rate emerging clusters and the most negative growth rate declining cluster, all classified with high confidence. Both Mann-Kendall and the Negative Binomial regression agree on direction for all three. For these three clusters, Mann-Kendall's own significance does not survive Benjamini-Hochberg correction (raw $p = 0.0355$, FDR corrected $p = 0.1944$), while the Negative Binomial growth rate remains significant after correction for all three. The high confidence label reflects agreement at the nominal significance level, not statistical significance after multiple-testing correction. Thus, these clusters should not be interpreted as having independently confirmed Mann-Kendall significance after FDR correction. Figure 4.24 shows each of these three clusters' raw yearly item counts with the linear and polynomial fits overlaid.

Table 4.12: Case study clusters, strongest trend signal

Cluster label	Trend category	Confidence	Mann-Kendall result	NB growth rate	Selected because
Coding Agent · Coding · Agent · Agentic Coding · AI Coding	Emerging	High	Increasing	699.35%	Top emerging by growth rate
Vibe · Vibe Coding · Coding · Context Vibe · Rise Vibe	Emerging	High	Increasing	490.06%	Second emerging by growth rate
View · Binding · View Binding · Fragment · Android	Declining	High	Decreasing	−78.52%	Most negative growth, declining anchor

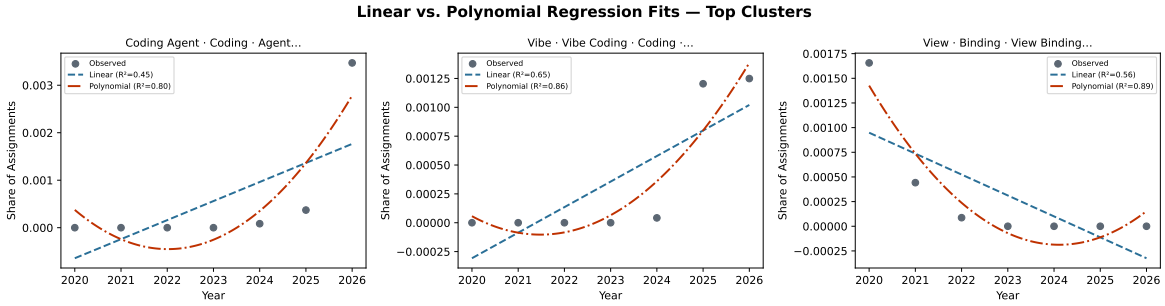


Figure 4.24: Yearly item counts with linear and polynomial fits, clusters selected for strongest trend signal

Table 4.13 shows three clusters chosen instead to illustrate where the classification is less certain. The first has a Negative Binomial growth rate of -5.03% per year, just past the fixed 5% threshold that separates declining from stable, and Mann-Kendall finds no significant trend for it ($p = 0.23$), so its declining label rests entirely on a growth rate close to the boundary that defines the category. The second shows a raw Negative Binomial p-value of 0.0138, nominally significant at the conventional 0.05 cutoff, that does not survive Benjamini-Hochberg correction ($p_{fdr} = 0.0561$), independent of Mann-Kendall, which never found a significant trend for this cluster either. The third is a stable cluster, included because the entire stable category is classified with low confidence, as reported above. Figure 4.25 shows the same, for each of these three clusters.

Table 4.13: Case study clusters, classification uncertainty

Cluster label	Trend category	Confidence	Mann-Kendall result	NB growth rate	Selected because
Browser · Chromium · Firefox · Chrome · Google Chrome	Declining	Low	No trend	−5.03%	Growth rate just past the 5% threshold
Kubernetes · Cluster · Kubernetes Cluster · Pod · Containerization	Declining	Medium	No trend	−12.24%	Raw NB significance lost after FDR correction
Agile · Extreme Agile · Agile Movement · Agile Magic · Classic Agile	Stable	Low	No trend	0.07%	Representative stable cluster

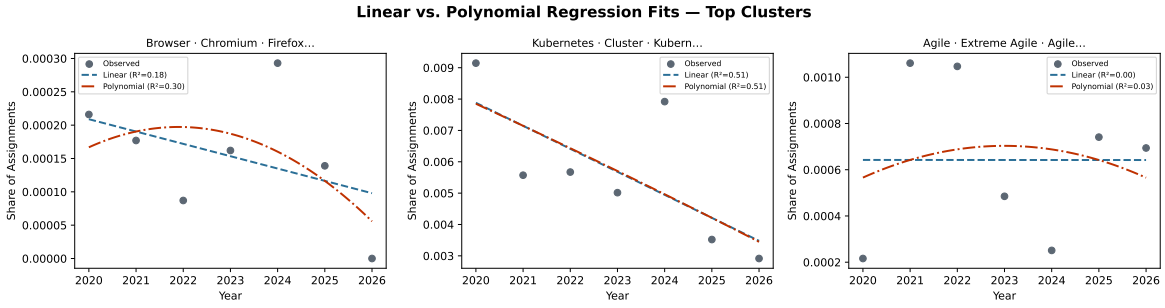


Figure 4.25: Yearly item counts with linear and polynomial fits, clusters selected to illustrate classification uncertainty

4.2 Discussion

Section 4.1 reported what the data shows. This section states what those results mean for RQ1 and RQ2, and how much confidence each answer actually deserves.

4.2.1 RQ1: Pipeline Feasibility

Direct answer. The pipeline extracts and clusters Software Engineering topics well. Manual review found accuracy high on both platforms, notably higher on Stack Exchange than on YouTube, and the clustering results tell a consistent story. Cluster structure clearly beats what a random partition of the same points would produce, membership assignment is confident on average, and a strong majority of manually reviewed clusters were judged coherent. Neither extraction nor clustering is the weak link the other has to compensate for.

Why this matters beyond the two platforms. This result speaks to the motivation behind using informal developer discourse in the first place. Transcripts and forum posts are worth mining precisely because an automated pipeline can extract usable structure from them, not only because they are abundant and cheap to collect.

What qualifies that confidence. Two caveats run through every number above. First, both manual checks, extraction accuracy and cluster coherence, were carried out by a single annotator with no independent second rater. This does not make the percentages wrong, but they should be read as one person's careful judgment, not an externally confirmed result. Second, the extraction step has a small number of concrete, named weaknesses worth stating plainly. The zero topic discards were dominated by a parsing failure rather than a genuine absence of relevant content, the model extracted something usable but failed to return it in the expected format, which is an encouraging finding since it points to a fixable formatting problem rather than a hard ceiling on what the model can find in this material. Separately, the cap on topics per item was reached often enough on both platforms, meaningfully more often on YouTube, that some items likely had more topics worth extracting than the cap allowed, understating topic diversity for those specific items.

On the clustering side, the one soft spot worth carrying forward is the noise ratio. More than a third of topic labels were not confidently placed in any cluster, and while the clusters that did form are demonstrably sound, this share of unplaced labels is not negligible and should be kept in mind wherever cluster based counts are used downstream.

Taken together, the evidence supports a confident yes to RQ1. The caveats above narrow how far that confidence extends, they do not undercut it.

4.2.2 RQ2: Topic Frequency and Evolution

Direct answer. This corpus shows the field's attention is not static. Roughly half of all clusters are classified as emerging and about a third as declining, leaving a smaller remainder classified as stable and a small residual group where the data was too sparse to

classify at all. This is a corpus dominated by directional movement rather than by topics holding steady.

The qualification that matters immediately. That movement rests unevenly on the underlying statistical evidence. A large share of both the emerging and the declining classifications is carried by Mann-Kendall's direction alone, without independent confirmation from the Negative Binomial fallback, and once correction for multiple testing is applied, the primary test's raw significance mostly does not survive it. The honest summary is that this corpus shows widespread directional change across Software Engineering topics between 2020 and 2026, but only a modest fraction of that change is backed by the strongest available evidence.

What the static landscape suggests. Before any trend is applied, the shape of the corpus already points toward a mature, recurring topic base rather than one dominated by a handful of short lived spikes. Attention is only moderately concentrated, a majority share of clusters together account for most of the item assignments, and most clusters remain active across nearly the entire collection window rather than appearing briefly and disappearing. The yearly Shannon entropy values confirm this at a corpus wide level, staying well above the low end of the scale in every year.

Confidence, stated honestly. Confidence in the individual trend classifications is mixed. Only a minority of emerging and declining clusters are backed by high confidence signals, most rest on medium or low confidence evidence instead, and the stable category is entirely low confidence by construction, since a stable label there reflects an absence of detected movement rather than positive evidence of steadiness. A further small group of clusters could not be classified at all, because too little year to year data existed to produce a usable growth rate, and this is reported as its own undetermined category rather than folded into stable, since the two situations mean different things.

Volatility adds a further layer of nuance. Year over year volatility exceeds coefficient of variation for nearly every cluster, so a cluster's path from one year to the next is typically jumpier than its overall spread around the mean would suggest on its own. This gap does not line up with the trend category assigned to a cluster, emerging, stable, and declining clusters all sit together in both the tightly packed region of low volatility and the sparse tail of extreme values. A cluster can carry a confident emerging or declining label and still move unpredictably from one year to the next, which is a further reason to treat those labels as summaries of overall direction rather than a claim that the underlying series moves smoothly.

The case studies make this concrete. The two strongest emerging clusters both trace back to the same broader shift, the rise of Artificial Intelligence (AI) assisted and agentic coding tools, an unsurprising finding given how visible that shift has been across the industry over this period. The strongest declining cluster tracks a specific piece of platform tooling losing ground as attention moves elsewhere in the same ecosystem, also unsurprising. The representative stable cluster shows essentially no detectable movement either way, which

is exactly why the stable category carries no confident claim of its own, a stable label is the absence of evidence for change, not evidence for the absence of change.

Taken together, RQ2's answer is that this corpus does show clear directional movement in Software Engineering topics over the collection window, concentrated in a recurring core of long lived topics rather than scattered across short lived ones, but the strength of that finding varies a great deal by cluster, and the emerging and declining labels should be treated as a starting point for further scrutiny rather than a settled fact for every cluster they cover.

4.3 Threats to Validity

This section names the ways the design of the pipeline and the corpus itself could limit how far the results in Section 4.1, and their interpretation in Section 4.2, can be trusted or generalized.

4.3.1 Internal Validity

Single annotator bias. Both manual evaluations in this thesis, extraction accuracy and cluster coherence, were carried out by a single annotator, the author of this thesis, with no independent second rater. A single annotator's judgment can be internally consistent without being correct, since one person can apply a personal standard for what counts as accurate or coherent without external confirmation that the standard matches how another reader would judge the same items. This thesis cannot rule that out, and every percentage from either manual check should be read with this in mind.

Fixed threshold constants in trend classification. The rule assigning each cluster to emerging, stable, declining or undetermined depends on two hand set constants rather than values learned from the data, the Mann-Kendall significance cutoff and the minimum growth rate magnitude that separates a real trend from stable. Both are conventional, reasonable choices, but they are still choices, and a cluster whose true growth rate sits close to either boundary could be classified differently under a slightly different threshold.

Polynomial fit as a comparison of unequal models. A degree two polynomial nests the linear model, so it will always match or exceed the linear fit's explanatory power regardless of whether the added curvature reflects a genuine pattern or is simply fitting noise in a short series. This is precisely why Mann-Kendall, rather than a comparison between polynomial and linear fits, was chosen in Section 3.7 as the primary significance test behind the classification rule, since it avoids depending on a comparison between models of unequal complexity in the first place.

The locally hosted extraction model. This thesis deliberately uses a locally hosted Llama 3.1 8B model for topic extraction rather than a larger, frontier hosted model, chosen for cost, reproducibility and independence from a third party API. Smaller open weight models are generally weaker at nuanced or implied interpretation, so cases where a discussed topic is implied rather than stated outright are more likely to be missed or misclassified by an 8B model than a larger one. Some share of the extraction accuracy result, and of the errors found in the failure analysis, may reflect this specific model choice rather than a limitation of using an LLM for extraction in general.

Fixed topic caps per item. The extraction step enforces a fixed maximum number of topics per item, described in Section 3.6. Both platforms reached that cap often enough, YouTube noticeably more than Stack Exchange, that the cap cannot be treated as a ceiling that rarely binds in practice. This means the pipeline likely under reports topic diversity for

a meaningful share of items on both platforms, understating cluster sizes and item counts throughout Section 4.1.3 rather than representing only a minor edge case.

Non-zero extraction temperature. Topic extraction ran at a temperature of 0.2 rather than a fully deterministic setting, so the same item could in principle receive a slightly different set of extracted topics on a different run. A stability check across a repeated run of the same items was considered but not carried out, so the practical impact of this setting remains an untested assumption rather than a confirmed source of stability.

4.3.2 External Validity

Corpus composition. The corpus is not evenly distributed across its sources. At the item level, the split between YouTube and Stack Exchange is fairly close, but Stack Exchange itself is unevenly split between its two constituent sites, one site contributes a clear majority of Stack Exchange items over the other. Any claim in this thesis that reads as being about Software Engineering discourse broadly, rather than about the specific mix of sources actually collected, should be read against this imbalance.

Sparse years in the collected data. The year 2026 has comparatively low item counts at every pipeline stage, a fact about how much usable data exists for this year rather than about how much the field discussed Software Engineering topics during that time. Any trend classification that leans heavily on a cluster's 2026 counts is built on a thinner evidence base than one drawn mainly from years with fuller collection, and generalization from this thesis's emerging or declining classifications about the field's most recent activity should treat this year as less reliable evidence than the rest of the window.

Generalizability beyond the specific sources sampled. The topics identified throughout this thesis come from a specific set of YouTube channels and two specific Stack Exchange sites, not from Software Engineering discourse as a whole. Even a pipeline that performs extraction and clustering perfectly can only describe the slice of discourse actually present in the sources it was given, and the trend classifications should be read as findings about this corpus's coverage of the field, not as a claim about the field in general.

5

Related Work

Understanding how software engineering research aligns with industry practice and how developer interests evolve over time, requires looking upon several distinct research areas. This chapter contextualizes our work within the broader literature across six key thematic areas: the research-practice gap, mining informal discourse and gray literature, topic extraction using Large Language Models, prompt engineering strategies, embedding and density-based clustering, and statistical time-series analysis for topic trends. Each section summarizes the current state of research and explicitly highlights how the methodology and scope of this thesis extend prior work.

Research-practice gap in Software Engineering

There is an ongoing, discussion in Software Engineering research regarding the practical relevance and applicability of academic contributions to industrial practice. As early as the 1990s, Glass [27] raised concerns that Software Engineering research was not sufficiently connected to industrial needs and often ignored practical application. This debate has persisted, with several authors describing a persistent gap between researchers and practitioners, often driven by differing goals, incentives, and evaluation criteria [9, 22]. Such differences can result in academic contributions that are theoretically sound but difficult to transfer to "real-world" development environments.

Empirical studies have sought to investigate this mismatch more systematically. For instance, Lo et al. [39] conducted a large scale survey where practitioners rated the relevance of research papers. Their findings revealed that while many papers were perceived positively, a significant portion was rated as having limited practical applicability or even being "unwise" for industry adoption. This suggests that perceptions of relevance vary significantly depending on the specific topic and context.

More recent work has continued to reflect on the applicability of Software Engineering research outcomes, emphasizing the critical importance of evaluating whether proposed methods and tools can realistically be transferred to and adopted by industry settings [71]. Overall, these discussions highlight that research–practice alignment remains a vital and evolving challenge for the Software Engineering community.

Gray literature and Mining Informal Developer Discourse

Within this broader body of work, Stack Overflow (and Stack Exchange sites more generally) is a really popular studied informal source. Its question-and-answer format, combined with community signals like votes and view counts, makes it comparatively easy to mine at scale. A number of studies have applied LDA to large samples of Stack Overflow posts in order to identify recurring topics, characterize emerging technologies, or track how developer concerns shift over time [3, 63]. These studies are arguably the closest methodological predecessors to the clustering and trend part of our thesis's pipeline, even though they rely on a different topic extraction technique.

YouTube, by comparison, is a much less explored data source for this kind of research. The handful of existing studies that use YouTube for Software Engineering purposes have primarily focused on mapping educational video content, tutorial structures, or online training materials [55, 64]. To the best of our knowledge, no prior work has used YouTube content as a lens on evolving practitioner interests in the same way that Stack Overflow has been used. This relative scarcity is itself worth noting, since it strengthens the case for treating YouTube as a data source for this kind of study.

This thesis builds directly on this line of work but differs from it in two ways. First, rather than treating Stack Exchange sites and YouTube as separate case studies, we combine both sources within a single topic extraction pipeline, which lets us compare how a given topic is discussed across a structured Q&A platform and a long-form, conversational one. Second, where the closest prior studies rely on LDA, this thesis replaces that step with LLM-based extraction, which is discussed in the next section.

Large Language Models for Topic Extraction

Automated topic extraction has been widely used to analyze large collections of textual data. Earlier work commonly relied on probabilistic models such as Latent Dirichlet Allocation [5] to identify latent themes in for example issue trackers, mailing lists or developer forums [3, 7, 14, 62, 63]. These studies showed that topic modeling with LDA can help summarize large text and reveal recurring technical concerns within software projects.

More recent work has replaced statistical methods like LDA with Large Language Models in order to extract the topic of a given document [38].

LLM topic extraction is framed as a suitable replacement for classical topic modeling methods like LDA, arguing that it is a well suited approach due to its semantic understanding and flexibility through prompts [49].

Getting an LLM to generate a topic label from a document consistently is itself an active area of research, the so called Prompt Engineering, it contains several prompting techniques ranging from zero-shot instructions to few-shot and role-based techniques [58, 60], and dedicated work exists on getting models to follow a consistent output format, such as a fixed label or a well-formed JSON object, which matters for any pipeline that processes model output automatically without any human intervention [73].

This thesis follows the approach of using a Large Language Model to extract topics instead of LDA, but applies it to informal and unstructred Software Engineering texts. We apply the same underlying extraction idea across two structurally very different sources (short technical Q&A posts and long, unstructured/conversational video transcripts), which require specific prompt design for each scenario instead of a single generic prompt.

Embedding-based and density based clustering in topic analysis

Once topics have been extracted as short text labels, a separate step is needed to group semantically related labels together (e.g., "memory leak" and "memory management issue"). A widely used family of approaches for this kind of problem, popularized under the name BERTopic [30], combines dense sentence embeddings with a density-based clustering algorithm, typically HDBSCAN [8], to group documents, and then derives a human-readable topic representation for each resulting cluster after the fact using a class-based variant of TF-IDF (Class-based Term Frequency-Inverse Document Frequency (c-TF-IDF)) [30, 57]. The BERTopic approach: embed, cluster, then label has since been adopted and extended in a range of follow-up work across different domains and text types [37, 59].

The clustering algorithm underlying the BERTopic approach, HDBSCAN, extends the classical DBSCAN algorithm [16] by building a hierarchy of clusters across varying density thresholds and extracting the most stable clusters from that hierarchy, rather than requiring a single global density parameter [8]. This makes it well suited to text clustering settings where cluster sizes and densities vary substantially and not every point necessarily belongs to a meaningful cluster, both of which apply to the topic labels considered in this thesis.

This thesis borrows some parts from the BERTopic approach like the embedding and HDBSCAN usage but departs from it in an important way. In the standard BERTopic pipeline, clustering is performed directly on embedded documents, and the human-readable topic label is only derived afterwards, from the words most characteristic of each resulting cluster via c-TF-IDF. In this thesis, the order is effectively reversed: the human-readable topic label is obtained directly from the LLM extraction step described in the previous sections, and sentence embeddings are used purely to measure the semantic similarity between these already labeled topics for the purpose of clustering them. This distinction is, as far as we are aware, a genuine point of methodological novelty relative to the standard BERTopic approach and its immediate descendants.

Beyond this methodological difference, using an LLM directly for topic extraction is also motivated by evidence that LLM-based extraction can outperform BERTopic pipelines in terms of topic quality and interpretability, which gives us additional confidence in obtaining the topic label directly from an LLM [31, 50, 66].

Trend and time-series analysis of topic popularity

The final part of the pipeline is concerned with detecting whether the prevalence of a given topic cluster changes meaningfully over time. The Mann-Kendall test, a non-parametric, rank-based test for a monotonic trend in a time series, was originally proposed by Mann

and later formalized by Kendall [36, 41]. It has since been applied well beyond its original statistical setting, including in bibliometric studies that track the rise and fall of research topics over time and in technology-adoption studies that characterize how interest in a given tool or framework evolves [12, 42]. Its main appeal in these settings is that it does not assume any particular functional form for the underlying trend and is comparatively robust to outliers, which matters when working with short, noisy series such as yearly topic counts.

Complementary to the Mann-Kendall trend test, low-degree polynomial regression is a standard time-series technique for capturing curvature in a trend, that is, for detecting whether a series is accelerating, slowing down, or reversing direction, rather than only whether it is increasing or decreasing overall [69]. Instead of using only simpler linear trend estimates, multiple polynomials of different degrees are used, since a single method on its own tends to miss either the overall direction (in the case of curve-fitting alone) or the curvature (in the case of a linear fit alone).

Negative Binomial regression offers a way to test for a trend directly on the count data itself while explicitly accounting for overdispersion, that is, variance in excess of what a Poisson model assumes. Abraham et al. [1] apply this approach to short and sparse monthly counts of reported artificial intelligence incidents, fitting a Negative Binomial generalized linear model with time as a predictor to estimate the direction and significance of a trend, after finding that a Poisson specification was severely violated due to substantial overdispersion in their data. Their setting, testing a small number of counts across a limited number of time periods for a change over time, closely parallels the setting of this thesis.

This thesis follows this general idea of combining complementary trend detection methods but applies it in a setting with notably few data points per series, in our case only seven (2020 -2026) yearly observations, which is shorter than the series typically analyzed in the bibliometric studies referenced above. Rather than relying on a single trend detection method, we combine the Mann-Kendall trend test with Negative Binomial regression. This combination of the two methods, specifically applied to LLM-extracted and clustered topic labels, is, to our knowledge, not something prior work in this space has done.

6

Concluding Remarks

6.1 Conclusion

Software Engineering research and industry practice do not always move together, and closing that gap starts with knowing what practitioners are actually talking about, at a scale no manual reading effort can keep up with. This thesis set out to build a pipeline that could observe that discourse directly, extracting and tracking Software Engineering topics from informal, high volume sources like YouTube and Stack Exchange, without relying on a fixed topic list decided in advance and without the labor cost that has historically limited this kind of analysis.

The main contribution of this thesis is the pipeline itself, together with the empirical evaluation of its feasibility. Across extraction, clustering, and trend classification, the results provide evidence that the proposed approach can produce useful and interpretable topic structures from informal Software Engineering discourse. The topic labels assigned by the LLM showed substantial agreement with the manual evaluation, the resulting clusters were generally coherent, and the trend analysis provided a systematic way of characterizing changes in topic prevalence over time. These results support the use of the pipeline as a practical approach for exploratory analysis of large-scale developer discourse, while the limitations discussed in this thesis mean that its outputs should not be interpreted as definitive measurements of practitioner attention.

The specific trends found here, including a corpus-wide shift toward AI-assisted and agentic coding tools and a recurring set of long-lived core topics, describe patterns of topic prevalence and activity within the sampled sources over one particular window in time. Software Engineering discourse will keep moving after this thesis is finished, today's emerging topic is next year's stable one, and today's stable topic could be next year's decline. What does not expire is the pipeline that produced this snapshot. The specific trends identified in this study will change as Software Engineering discourse evolves. The broader value of the pipeline is that it provides a reusable framework for analyzing those changes in future collections of data.

That reusability is where this thesis's broader significance lies. For researchers, it offers a repeatable way to check whether a research agenda still tracks what practitioners actually care about, rather than relying on citation counts or conference themes as a proxy. For industry, it offers a way to see which tools, practices, and technologies are gaining or

losing developer attention, grounded in what practitioners say themselves rather than in vendor reporting or survey samples. And for the field's ongoing conversation about the gap between research and practice, it offers a concrete instrument for measuring that gap over time, rather than only describing it. The pipeline is the answer this thesis set out to build. What it finds each time it runs will keep changing. The pipeline that finds it does not.

6.2 Future Work

The pipeline built for this thesis performs well on the corpus it was given, but every choice made to get it there, which model to use, how many manual checks to run, where to set a threshold, opens a direction worth exploring further rather than closing off. The opportunities below build directly on what this thesis established, they are not items left unfinished.

Confirming the manual checks with a second rater. Every accuracy and coherence judgment in this thesis came from a single annotator. The natural next step is an inter rater agreement study, a second annotator scoring the same samples independently, which would turn the strong accuracy and coherence numbers already found here into an externally confirmed result rather than one person's careful judgment. Given how well the pipeline already performed under a single rater, there is good reason to expect this would strengthen the finding rather than complicate it.

Testing whether a larger model changes the picture. This thesis deliberately chose a small, locally hosted model for reasons of cost and time constraints, and it delivered strong extraction quality even so. A natural follow up is running the same extraction step with a larger, frontier hosted model on the same corpus, to see how much headroom that specific choice left on the table, and whether the gap is worth the added cost and reduced reproducibility for a given use case. This also opens the door to the comparison against classical topic modeling that motivated part of this thesis's original goal, a direct benchmark against an LDA based baseline would settle how much of this pipeline's advantage comes from using an LLM at all, versus from the specific model size chosen.

Letting the trend thresholds learn from the data. The constants that separate emerging from stable from declining were set by hand in this thesis, reasonable choices, but choices nonetheless. Future work could explore data driven alternatives, thresholds calibrated against the actual volatility and growth rate distributions this thesis already characterized, rather than fixed in advance. This would let the classification rule adapt to however noisy or smooth a future corpus turns out to be, instead of applying the same cutoff regardless.

Growing the pipeline beyond two platforms. YouTube and Stack Exchange offered two genuinely different registers of practitioner discourse, but they are far from the only ones. Extending data collection to sources like Reddit's developer communities, Hacker News, GitHub issue discussions, or developer newsletters would let the same pipeline compare practitioner attention across an even wider slice of where that attention actually

lives, and would help correct the platform imbalance noted as a limitation of the current corpus.

Turning a one time analysis into a standing one. The most direct way to act on this thesis's central claim, that the pipeline outlasts its own results, is to actually run it again. A natural next step is scheduling the pipeline to re ingest fresh data on a recurring basis and re run the full trend analysis each time, turning this thesis's snapshot into a living, periodically refreshed view of practitioner attention. Comparing successive snapshots against each other would also open a new question this thesis never had the data to ask, not just which topics are emerging or declining within one window, but how consistent a topic's trajectory is across multiple independent windows over time.

Following up on individual topics with a closer read. The trend classification in this thesis works at the scale of thousands of clusters, which is exactly its strength, but it also means every finding is necessarily a wide angle one. A promising follow up is picking a small number of the most confidently classified emerging or declining clusters and studying them qualitatively, reading a sample of the actual transcripts and posts behind the numbers, to understand not just that a topic is rising but why practitioners are talking about it more, information the current pipeline was never designed to surface on its own.

A

Appendix

This appendix collects supplementary material that supports the methodology and results presented in the main text. It includes reference material such as full data collection lists, along with additional metrics and statistics that are not essential to the core argument but provide further detail for readers interested in the underlying data and computations

A.1 YouTube Channel List

The 107 channels were selected manually. Three criteria guided the selection: the channel uploads content in English, it has published at least one video from 2020 onwards, and its content is about software engineering. Candidate channels were identified through the YouTube Data API using the search terms "Software Engineering", "Software Engineering podcasts", and "Computer science". A small number of conference channels were included as well. Table A.1 lists every YouTube channel handle included in the data collection described in Section 3.4.2.

Table A.1: List of YouTube channel handles included in the data collection corpus

Handle	Handle
@137-podcast	@AdventuresInDevOps
@agileandtechinsweden266	@agilitymatrix
@AIDevopsPodcast	@AliKhanPodcastt
@AWSDevelopersPodcast	@backendbanterfm
@betterenggmgmt	@Beyond-Deploy
@BeyondCoding	@bigboxSWE
@ChainOfThoughtAI	@Changelog
@codestories	@CodingChats
@continuousdeliverypodcast	@cultureofcode
@dev-ness	@devbricksinc
@DevCoaches	@devdialog
@developertea	@DevelopYourselfPodcast
@develpreneur	@devhopspodcast9535

Continued on next page

Table A.1 – continued from previous page

Handle	**Handle**
@devshopstories	@digitaledgepodcast
@disseminatepodcast	@DonTheDeveloper
@EngineeringQuality	@engxspace
@fragmentedpodcast	@freecodecamp
@FrontEndHH	@Future-of-tech-podcast
@glich.stream	@GoogleDevelopers
@GoogleSREProdcast	@GOTO-
@GRCEngineer	@HealthyDev
@inspireCasts25	@Interpolation-podcast
@jamesblueai	@javascriptlifestylespodcas4086
@Jay-Shah	@JOSSCast
@junior_jobs	@kopioverflow6866
@LadybugPodcast	@LEADpodcast
@LennysPodcast	@lets-talk-agile
@localfirstfm	@lvlupeng
@mapforengineers	@mobmentalityshow
@mobycast8407	@ModernSoftware-me
@ModernSoftwareEngineeringYT	@mohammadfraz
@NDC	@NETRocks-pr4iv
@NTGen1Podcasts	@podcast-as-code
@PodcastforSoftwareTesters	@pragmaticengineer
@programmingwithtsatsu	@SajjaadKhader
@scalingdevtools	@scalingtechpodcast
@ScrumMasterToolboxPodcast	@SecOopsPodcast
@SiliconValleyGirl	@SmartAndSnark
@SoftwareDaily	@SoftwareLeadersUncensored
@SoftwareMeetsBusiness	@SoftwareUnscripted
@SpotRoboticsPodcast	@StackedSessions-Podcast
@SuperDataScienceWithJonKrohn	@syntaxfm
@tatacpodcast	@TechandTestPodcast
@TechnicalSuneja	@tecladocode
@The-Shivam-Yadav9	@theagilestanduppodcast8059
@thebossrodpodcast	@TheCoderCareer
@thedevshortpodcast9784	@TheDigitalNativePodcast
@TheHalfStacks	@TheJoshuaChrisPodcast
@TheKubernetesPodcast	@TheModernEngineerPodcast
@ThePanelPodcast	@TheProgrammingPodcast
@thesystempodcast	@theundefinedpodcast5818

Continued on next page

Table A.1 – continued from previous page

Handle	Handle
@ThrivingInEngineeringPodcast	@TPRMPodcast
@weboftomorrowpodcast6422	@weeklydevtips
@WorstPractices	

a.2 Polynomial Regression Fit Details

The fit is performed on mean centered years, that is, on $x = \text{year} - \overline{\text{years}}$ rather than on the raw calendar year, which removes the correlation between x and x^2 that would otherwise make the fitted coefficients numerically unstable. The same centering, using the same mean year, must be applied whenever the resulting curve is plotted against the original calendar years.

The fit is reported as coefficients $[a, b, c]$ for $ax^2 + bx + c$, together with its own R^2 value, which can be compared against the linear fit's R^2 to see whether curvature meaningfully improves the description of the data. The sign of the leading coefficient a determines the reported concavity of the trend:

- $a > 10^{-8}$: concave up (for example, decline followed by recovery, or accelerating growth)
- $a < -10^{-8}$: concave down (for example, growth that slows down, or a topic that rises and then falls again)
- otherwise: near linear, meaning the quadratic term does not meaningfully change the fit compared to the straight line

Because a series has only seven points, the degree of the fitted polynomial is automatically reduced if a cluster's series does not contain enough distinct points to support a quadratic fit, for example if the cluster is missing observations in some years.

a.3 Cross-Platform Cluster Overlap

A cluster is classified as shared if it contains at least one YouTube item and at least one Stack Exchange item, and as belonging exclusively to a single platform otherwise. Of the 2,470 clusters found by the pipeline, 1091 contain items from YouTube only, 29 contain items from Stack Exchange only, and 1,350 are shared between both platforms, corresponding to an overlap ratio of 54.7%.

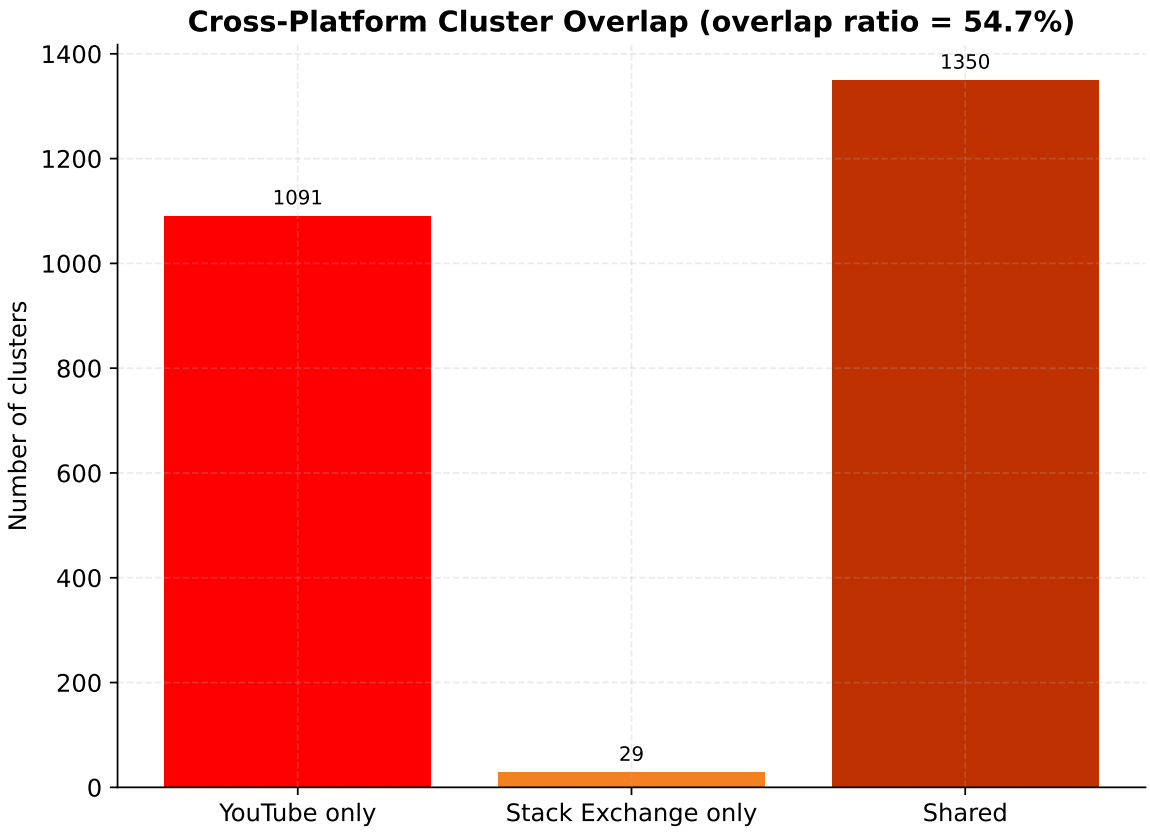


Figure A.1: Distribution of item clusters by platform origin. Bars show the number of clusters containing items from YouTube only, Stack Exchange only, or both platforms, out of 2470 total clusters

a.4 Trend Test P-Value Distributions

This section shows, for reference, the full distribution of p-values produced by the linear regression, Mann-Kendall, and Negative Binomial trend tests across all clusters, complementing the significant and non-significant cluster counts already reported in Section 4.1.3.2.

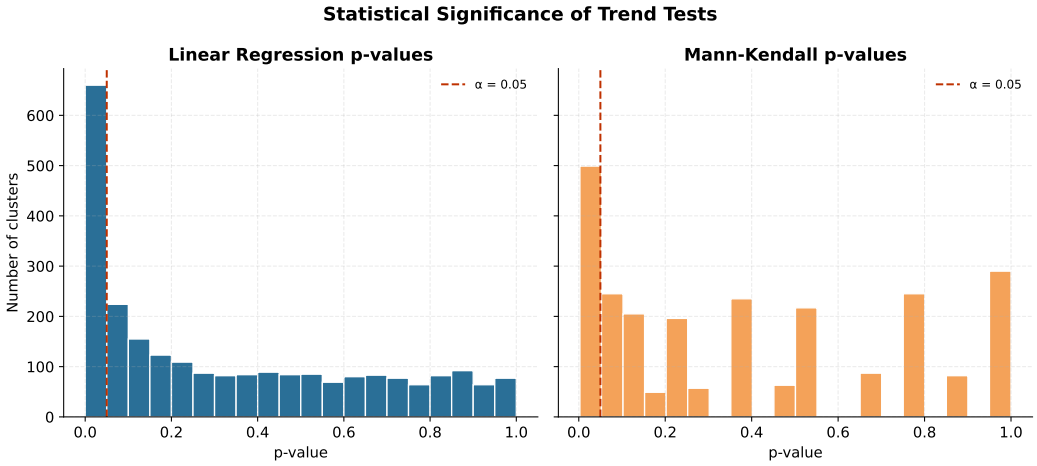


Figure A.2: Distribution of linear regression and Mann-Kendall p-values across all clusters

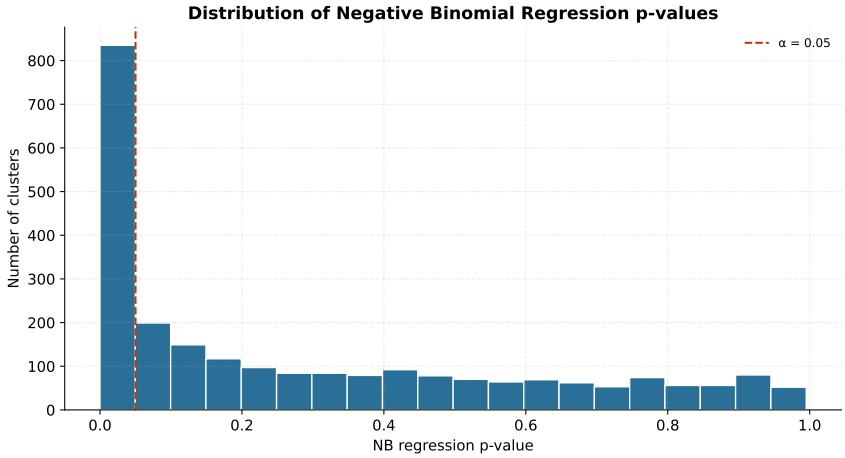


Figure A.3: Distribution of Negative Binomial regression p-values across all clusters with a usable fit

Statement on the Usage of Generative Digital Assistants

For this thesis the following generative digital assistant were used: Google Gemini[1] and Claude Sonnet 5[2]; they were used as an aid to identify synonyms, to structure ideas and to refine vocabulary selections (e.g., transitions and advanced terminology that are only used in formal papers). They were always used through their web interface.

Claude Sonnet 5[2] was also used for code completion tasks. It was mainly used to write helper functions, like to parse JSON files, to create figures or to load data in a specific way. It was also used to document code. It was used always through its web interface.

We are aware of the dangers of using these tools blindly, that was never the case, these tools were used sensibly and with critical thinking. They were just an aid.

[1] https://gemini.google.com
[2] https://claude.ai

Bibliography

[1] Sophia Abraham, Taiye Chen, Cyril Chhun, Giovanna Jaramillo-Gutierrez, Simon Mylius, Sayash Raaj, Peter Slattery, and Sean McGregor. *AI Incident Monitoring through a Public Health Lens*. 2026. arXiv: 2604.19914 [cs.CY]. URL: https://arxiv.org/abs/2604.19914.

[2] Mohammad Akhavan Anvari, Dara Rahmati, and Sunil Kumar. "t-Distributed stochastic neighbor embedding." In: *Dimensionality Reduction in Machine Learning*. Elsevier, 2025, pp. 187–207.

[3] Anton Barua, Stephen W. Thomas, and Ahmed E. Hassan. "What are developers talking about? An analysis of topics and trends in Stack Overflow." In: *Empirical Softw. Engg.* 19.3 (June 2014), 619–654. ISSN: 1382-3256. DOI: 10.1007/s10664-012-9231-y. URL: https://doi.org/10.1007/s10664-012-9231-y.

[4] Sarah Beecham, Pádraig O'Leary, Sean Baker, Ita Richardson, and John Noll. "Making Software Engineering Research Relevant." In: *Computer* 47.4 (2014), pp. 80–83. DOI: 10.1109/MC.2014.92.

[5] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. "Latent dirichlet allocation." In: *J. Mach. Learn. Res.* 3.null (Mar. 2003), 993–1022. ISSN: 1532-4435.

[6] Tom B. Brown et al. *Language Models are Few-Shot Learners*. 2020. arXiv: 2005.14165 [cs.CL]. URL: https://arxiv.org/abs/2005.14165.

[7] Joshua Charles Campbell, Abram Hindle, and Eleni Stroulia. "Latent Dirichlet allocation: extracting topics from software engineering data." In: *The art and science of analyzing software data*. Elsevier, 2015, pp. 139–159.

[8] Ricardo J. G. B. Campello, Davoud Moulavi, and Joerg Sander. "Density-Based Clustering Based on Hierarchical Density Estimates." In: *Advances in Knowledge Discovery and Data Mining*. Ed. by Jian Pei, Vincent S. Tseng, Longbing Cao, Hiroshi Motoda, and Guandong Xu. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 160–172. ISBN: 978-3-642-37456-2.

[9] Andrea Capiluppi. "Dear researchers: Turning industry into a laboratory — The UnICo experience." In: *Journal of Systems and Software* 228 (2025), p. 112495. ISSN: 0164-1212. DOI: https://doi.org/10.1016/j.jss.2025.112495. URL: https://www.sciencedirect.com/science/article/pii/S0164121225001633.

[10] Banghao Chen, Zhaofeng Zhang, Nicolas Langrené, and Shengxin Zhu. "Unleashing the potential of prompt engineering for large language models." In: *Patterns* 6.6 (2025), p. 101260. ISSN: 2666-3899. DOI: 10.1016/j.patter.2025.101260. URL: http://dx.doi.org/10.1016/j.patter.2025.101260.

[11] Aakanksha Chowdhery et al. *PaLM: Scaling Language Modeling with Pathways*. 2022. arXiv: 2204.02311 [cs.CL]. URL: https://arxiv.org/abs/2204.02311.

[12] Mario Coccia and Saeed Roshani. "Evolution of topics and trends in emerging research fields: multiple analyses with entity linking, Mannâ€"Kendall test and burst methods in cloud computing." In: *Scientometrics* 129.9 (2024), pp. 5347–5371. DOI: 10.1007/s11192-024-05139-4. URL: https://ideas.repec.org/a/spr/scient/v129y2024i9d10.1007_s11192-024-05139-4.html.

[13] Stefany Coxe, Stephen G. West, and Leona S. Aiken. "The Analysis of Count Data: A Gentle Introduction to Poisson Regression and Its Alternatives." In: *Journal of Personality Assessment* 91.2 (2009). PMID: 19205933, pp. 121–136. DOI: 10.1080/00223890802634175. eprint: https://doi.org/10.1080/00223890802634175. URL: https://doi.org/10.1080/00223890802634175.

[14] Andrea Di Sorbo, Sebastiano Panichella, Carol V. Alexandru, Junji Shimagaki, Corrado A. Visaggio, Gerardo Canfora, and Harald C. Gall. "What would users change in my app? summarizing app reviews for recommending software changes." In: *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*. FSE 2016. Seattle, WA, USA: Association for Computing Machinery, 2016, 499–510. ISBN: 9781450342186. DOI: 10.1145/2950290.2950299. URL: https://doi.org/10.1145/2950290.2950299.

[15] Trina Y Du. "Dimensionality reduction techniques for visualizing morphometric data: Comparing principal component analysis to nonlinear methods." In: *Evolutionary Biology* 46.1 (2019), pp. 106–121.

[16] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. "A density-based algorithm for discovering clusters in large spatial databases with noise." In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. KDD'96. Portland, Oregon: AAAI Press, 1996, 226–231.

[17] Stack Exchange. *Software Engineering Stack Exchange*. Website. Available online at https://softwareengineering.stackexchange.com; visited on July 20th, 2026. 2026.

[18] Stack Exchange. *Stack Exchange API*. Website. Available online at https://api.stackexchange.com; visited on July 20th, 2026. 2026.

[19] Stack Exchange. *Stack Exchange*. Website. Available online at https://stackexchange.com; visited on July 20th, 2026. 2026.

[20] Stack Exchange. *Stack Overflow*. Website. Available online at https://stackoverflow.com; visited on July 20th, 2026. 2026.

[21] Denis Federiakin, Dimitri Molerov, Olga Zlatkin-Troitschanskaia, and Andreas Maur. "Prompt engineering as a new 21st century skill." In: *Frontiers in Education* Volume 9 - 2024 (2024). ISSN: 2504-284X. DOI: 10.3389/feduc.2024.1366434. URL: https://www.frontiersin.org/journals/education/articles/10.3389/feduc.2024.1366434.

[22] Xavier Franch, Daniel Mendez, Andreas Vogelsang, Rogardt Heldal, Eric Knauss, Marc Oriol, Guilherme H. Travassos, Jeffrey C. Carver, and Thomas Zimmermann. "How do Practitioners Perceive the Relevance of Requirements Engineering Research?" In: *IEEE Transactions on Software Engineering* 48.6 (2022), pp. 1947–1964. DOI: 10.1109/TSE.2020.3042747.

[23] Vahid Garousi, Markus Borg, and Markku Oivo. "Practical relevance of software engineering research: synthesizing the community's voice." In: *Empirical Software Engineering* 25.3 (Mar. 2020), 1687–1754. ISSN: 1573-7616. DOI: 10.1007/s10664-020-09803-0. URL: http://dx.doi.org/10.1007/s10664-020-09803-0.

[24] Vahid Garousi, Michael Felderer, and Mika V. Mäntylä. "The need for multivocal literature reviews in software engineering: complementing systematic literature reviews with grey literature." In: *Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering*. EASE '16. Limerick, Ireland: Association for Computing Machinery, 2016. ISBN: 9781450336918. DOI: 10.1145/2915970.2916008. URL: https://doi.org/10.1145/2915970.2916008.

[25] Vahid Garousi, Michael Felderer, and Mika V. Mäntylä. *Guidelines for including grey literature and conducting multivocal literature reviews in software engineering*. 2018. arXiv: 1707.02553 [cs.SE]. URL: https://arxiv.org/abs/1707.02553.

[26] Vahid Garousi, Michael Felderer, Mika V. Mäntylä, and Austen Rainer. *Benefitting from the Grey Literature in Software Engineering Research*. 2019. arXiv: 1911.12038 [cs.SE]. URL: https://arxiv.org/abs/1911.12038.

[27] R.L. Glass. "The software-research crisis." In: *IEEE Software* 11.6 (1994), pp. 42–47. DOI: 10.1109/52.329400.

[28] Aaron Grattafiori et al. *The Llama 3 Herd of Models*. 2024. arXiv: 2407.21783 [cs.AI]. URL: https://arxiv.org/abs/2407.21783.

[29] Michael Greenacre, Patrick JF Groenen, Trevor Hastie, Alfonso Iodice d'Enza, Angelos Markos, and Elena Tuzhilina. "Principal component analysis." In: *Nature Reviews Methods Primers* 2.1 (2022), p. 100.

[30] Maarten Grootendorst. *BERTopic: Neural topic modeling with a class-based TF-IDF procedure*. 2022. arXiv: 2203.05794 [cs.CL]. URL: https://arxiv.org/abs/2203.05794.

[31] Weiqing He, Bojian Hou, Amy Zheng, Yanbo Feng, Ari Klein, Karen O'Connor, Shu Yang, Tianqi Shang, George Demiris, Graciela Gonzalez-Hernandez, and Li Shen. "Advanced topic modeling with large language models: analyzing social media content from dementia caregivers." In: *Innovation in Aging* 9.Supplement$_1$ (Dec. 2025), S38–S47. ISSN: 2399-5300. DOI: 10.1093/geroni/igaf120. eprint: https://academic.oup.com/innovateage/article-pdf/9/Supplement_1/S38/66134445/igaf120.pdf. URL: https://doi.org/10.1093/geroni/igaf120.

[32] Joseph M. Hilbe. *Negative binomial regression*. 2nd. Cambridge, 2014. ISBN: 9781107009400. DOI: 10.1017/CBO9780511973420.

[33] Huggingface. *"all-MiniLM-L6-v2 Sentence Transformer*. Website. Available online at https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2; visited on July 20th, 2026. 2021.

[34] Supakpong Jinarat and Ratchakoon Pruengkarn. "Enhancing Short Text Semantic Similarity Measurement Using Pretrained Word Embeddings and Big Data." In: *2024 5th International Conference on Big Data Analytics and Practices (IBDAP)*. 2024, pp. 63–66. DOI: 10.1109/IBDAP62940.2024.10689695.

[35] Adán José-García and Wilfrido Gómez-Flores. "A survey of cluster validity indices for automatic data clustering using differential evolution." In: *Proceedings of the Genetic and Evolutionary Computation Conference*. GECCO '21. Lille, France: Association for Computing Machinery, 2021, 314–322. ISBN: 9781450383509. DOI: 10.1145/3449639.3459341. URL: https://doi.org/10.1145/3449639.3459341.

[36] Maurice G. Kendall. *Rank Correlation Methods*. 4th. London: Charles Griffin, 1975.

[37] Nabila Khodeir and Fatma Elghannam. "Efficient topic identification for urgent MOOC Forum posts using BERTopic and traditional topic modeling techniques." In: *Education and Information Technologies* 30 (Sept. 2024), pp. 5501–5527. DOI: 10.1007/s10639-024-13003-4.

[38] Zongxia Li, Lorena Calvo-Bartolomé, Alexander Hoyle, Paiheng Xu, Alden Dima, Juan Francisco Fung, and Jordan Boyd-Graber. *Large Language Models Struggle to Describe the Haystack without Human Help: Human-in-the-loop Evaluation of Topic Models*. 2025. arXiv: 2502.14748 [cs.CL]. URL: https://arxiv.org/abs/2502.14748.

[39] David Lo, Nachiappan Nagappan, and Thomas Zimmermann. "How practitioners perceive the relevance of software engineering research." In: *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*. ESEC/FSE 2015. Bergamo, Italy: Association for Computing Machinery, 2015, 415–425. ISBN: 9781450336758. DOI: 10.1145/2786805.2786809. URL: https://doi.org/10.1145/2786805.2786809.

[40] Quenby Mahood, Dwayne Van Eerd, and Emma Irvin. "Searching for grey literature for systematic reviews: challenges and benefits." In: *Research Synthesis Methods* 5.3 (2014), pp. 221–234. DOI: https://doi.org/10.1002/jrsm.1106. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/jrsm.1106. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/jrsm.1106.

[41] Henry B. Mann. "Nonparametric Tests Against Trend." In: *Econometrica* 13.3 (1945), pp. 245–259. ISSN: 00129682, 14680262. URL: http://www.jstor.org/stable/1907187 (visited on 07/22/2026).

[42] Mauricio Marrone. "Application of entity linking to identify research fronts and trends." In: *Scientometrics* 122 (Nov. 2019), pp. 357–379. DOI: 10.1007/s11192-019-03274-x.

[43] Leland McInnes, John Healy, and James Melville. "Umap: Uniform manifold approximation and projection for dimension reduction." In: *arXiv preprint arXiv:1802.03426* (2018).

[44] Leland McInnes, John Healy, and James Melville. *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction*. 2020. arXiv: 1802.03426 [stat.ML]. URL: https://arxiv.org/abs/1802.03426.

[45] Meta. *meta-llama/Llama-3.1-8B-Instruct*. Accessed: 2026-08-18. 2024. URL: https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct.

[46] Gloriana Joseph Monko and Masaomi Kimura. "SS-DBSCAN: Epsilon estimation with stratified sampling for density-based spatial clustering of applications with noise." In: *2023 International Conference on Automation, Control and Electronics Engineering (CACEE)*. IEEE. 2023, pp. 72–76.

[47] Cristina Martinez Montes, Robert Feldt, Cristina Miguel Martos, Sofia Ouhbi, Shweta Premanandan, and Daniel Graziotin. *Large Language Models in Thematic Analysis: Prompt Engineering, Evaluation, and Guidelines for Qualitative Software Engineering Research*. 2025. arXiv: 2510.18456 [cs.SE]. URL: https://arxiv.org/abs/2510.18456.

[48] Davoud Moulavi, Pablo Andretta Jaskowiak, Ricardo Campello, Arthur Zimek, and Joerg Sander. "Density-Based Clustering Validation." In: Apr. 2014. DOI: 10.1137/1.9781611973440.96.

[49] Yida Mu, Chun Dong, Kalina Bontcheva, and Xingyi Song. *Large Language Models Offer an Alternative to the Traditional Approach of Topic Modelling*. 2024. arXiv: 2403.16248 [cs.CL]. URL: https://arxiv.org/abs/2403.16248.

[50] Yida Mu, Chun Dong, Kalina Bontcheva, and Xingyi Song. "Large Language Models Offer an Alternative to the Traditional Approach of Topic Modelling." In: *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*. Ed. by Nicoletta Calzolari, Min-Yen Kan, Veronique Hoste, Alessandro Lenci, Sakriani Sakti, and Nianwen Xue. Torino, Italia: ELRA and ICCL, May 2024, pp. 10160–10171. URL: https://aclanthology.org/2024.lrec-main.887/.

[51] Syed Naqvi, Shamsa Batool, Muzamil Ahmed, Hikmat Khan, and Muhammad Malik. "A Novel Approach for Building Domain-Specific Chatbots by Exploring Sentence Transformers-based Encoding." In: Oct. 2023, pp. 1–7. DOI: 10.1109/ICIT59216.2023.10335884.

[52] Cleo Pappas and Irene Williams. "Grey Literature: Its Emerging Importance." In: *Journal of Hospital Librarianship* 11.3 (2011), pp. 228–234. DOI: 10.1080/15323269.2011.587100. eprint: https://doi.org/10.1080/15323269.2011.587100. URL: https://doi.org/10.1080/15323269.2011.587100.

[53] Maša Pejović, Nikola Vasović, Antonio Cicchetti, and Robbert Jongeling. "Mapping educational software engineering content on YouTube." In: *Proceedings of the IEEE/ACM 48th International Conference on Software Engineering*. ICSE-SEET '26. New York, NY, USA: Association for Computing Machinery, 2026, 174–184. ISBN: 9798400724237. DOI: 10.1145/3786580.3786974. URL: https://doi.org/10.1145/3786580.3786974.

[54] Austen Rainer and Ashley Williams. "Using blog-like documents to investigate software practice: Benefits, challenges, and research directions." In: *J. Softw. Evol. Process* 31.11 (Nov. 2019). ISSN: 2047-7473. DOI: 10.1002/smr.2197. URL: https://doi.org/10.1002/smr.2197.

[55] Veland Ramadani, Hyrije Abazi, Lejla Abazi Bexheti, and Arbana Kadriu. "Investigating Trends in Learning Programming using YouTube Tutorials." In: *International Journal of Learning and Change* 1 (Jan. 2019), p. 1. DOI: 10.1504/IJLC.2019.10017809.

[56] Nils Reimers and Iryna Gurevych. *Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks*. 2019. arXiv: 1908.10084 [cs.CL]. URL: https://arxiv.org/abs/1908.10084.

[57] KAREN SPARCK JONES. "A STATISTICAL INTERPRETATION OF TERM SPECIFICITY AND ITS APPLICATION IN RETRIEVAL." In: *Journal of Documentation* 28.1 (Jan. 1972), pp. 11–21. ISSN: 0022-0418. DOI: 10.1108/eb026526. eprint: https://www.emerald.com/jd/article-pdf/28/1/11/1336479/eb026526.pdf. URL: https://doi.org/10.1108/eb026526.

[58] Pranab Sahoo, Ayush Kumar Singh, Sriparna Saha, Vinija Jain, Samrat Mondal, and Aman Chadha. *A Systematic Survey of Prompt Engineering in Large Language Models: Techniques and Applications*. 2025. arXiv: 2402.07927 [cs.AI]. URL: https://arxiv.org/abs/2402.07927.

[59] Sahil Sawant, Jinhong Yu, Kirtikumar Pandya, Chun-Kit Ngan, and Rolf Bardeli. "An Enhanced BERTopic Framework and Algorithm for Improving Topic Coherence and Diversity." In: *2022 IEEE 24th Int Conf on High Performance Computing Communications; 8th Int Conf on Data Science Systems; 20th Int Conf on Smart City; 8th Int Conf on Dependability in Sensor, Cloud Big Data Systems Application (HPCC/DSS/SmartCity/DependSys)*. 2022, pp. 2251–2257. DOI: 10.1109/HPCC-DSS-SmartCity-DependSys57074.2022.00332.

[60] Sander Schulhoff et al. *The Prompt Report: A Systematic Survey of Prompt Engineering Techniques*. 2025. arXiv: 2406.06608 [cs.CL]. URL: https://arxiv.org/abs/2406.06608.

[61] Margaret-Anne Storey, Leif Singer, Brendan Cleary, Fernando Figueira Filho, and Alexey Zagalsky. "The (R)Evolution of Social Media in Software Engineering." In: May 2014. DOI: 10.1145/2593882.2593887.

[62] Stephen W. Thomas. "Mining software repositories using topic models." In: *Proceedings of the 33rd International Conference on Software Engineering*. ICSE '11. Waikiki, Honolulu, HI, USA: Association for Computing Machinery, 2011, 1138–1139. ISBN: 9781450304450. DOI: 10.1145/1985793.1986020. URL: https://doi.org/10.1145/1985793.1986020.

[63] Christoph Treude and Markus Wagner. "Predicting Good Configurations for GitHub and Stack Overflow Topic Models." In: *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*. 2019, pp. 84–95. DOI: 10.1109/MSR.2019.00022.

[64] Nikola Vasovic and Masa Pejovic. *Towards a Systematic Mapping of YouTube videos on Software Engineering*. 2025.

[65] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL]. URL: https://arxiv.org/abs/1706.03762.

[66] Han Wang, Nirmalendu Prakash, Nguyen Khoi Hoang, Ming Shan Hee, Usman Naseem, and Roy Ka-Wei Lee. "Prompting Large Language Models for Topic Modeling." In: *2023 IEEE International Conference on Big Data (BigData)*. 2023, pp. 1236–1241. DOI: 10.1109/BigData59044.2023.10386113.

[67] Wenhui Wang, Hangbo Bao, Shaohan Huang, Li Dong, and Furu Wei. *MiniLMv2: Multi-Head Self-Attention Relation Distillation for Compressing Pretrained Transformers*. 2021. arXiv: 2012.15828 [cs.CL]. URL: https://arxiv.org/abs/2012.15828.

[68] Aasim Ayaz Wani. "Comprehensive review of dimensionality reduction algorithms: challenges, limitations, and innovative solutions." In: *PeerJ Computer Science* 11 (2025), e3025.

[69] Mike West and Jeff Harrison. "Polynomial Trend Models." In: *Bayesian Forecasting and Dynamic Models*. New York, NY: Springer New York, 1989, pp. 201–228. ISBN: 978-1-4757-9365-9. DOI: 10.1007/978-1-4757-9365-9_7. URL: https://doi.org/10.1007/978-1-4757-9365-9_7.

[70] Ashley Williams and Austen Rainer. "Toward the use of blog articles as a source of evidence for software engineering research." In: *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering*. EASE '17. Karlskrona, Sweden: Association for Computing Machinery, 2017, 280–285. ISBN: 9781450348041. DOI: 10.1145/3084226.3084268. URL: https://doi.org/10.1145/3084226.3084268.

[71] Titus Winters. "Thoughts on applicability." In: *Journal of Systems and Software* 215 (2024), p. 112086. ISSN: 0164-1212. DOI: https://doi.org/10.1016/j.jss.2024.112086. URL: https://www.sciencedirect.com/science/article/pii/S0164121224001316.

[72] Institute for Work Health. *What researchers mean by... Grey literature*. Website. Available online at https://www.iwh.on.ca/what-researchers-mean-by/grey-literature; visited on August 15, 2026. 2008.

[73] Congying Xia, Chen Xing, Jiangshu Du, Xinyi Yang, Yihao Feng, Ran Xu, Wenpeng Yin, and Caiming Xiong. *FOFO: A Benchmark to Evaluate LLMs' Format-Following Capability*. 2024. arXiv: 2402.18667 [cs.CL]. URL: https://arxiv.org/abs/2402.18667.

[74] Qinyuan Ye, Mohamed Ahmed, Reid Pryzant, and Fereshte Khani. "Prompt Engineering a Prompt Engineer." In: *Findings of the Association for Computational Linguistics: ACL 2024*. Ed. by Lun-Wei Ku, Andre Martins, and Vivek Srikumar. Bangkok, Thailand: Association for Computational Linguistics, Aug. 2024, pp. 355–385. DOI: 10.18653/v1/2024.findings-acl.21. URL: https://aclanthology.org/2024.findings-acl.21/.

[75] YouTube. *YouTube Data API*. Website. Available online at https://developers.google.com/youtube/v3; visited on July 20th, 2026. 2026.

[76] Tianyi Zhang, Faisal Ladhak, Esin Durmus, Percy Liang, Kathleen McKeown, and Tatsunori B. Hashimoto. *Benchmarking Large Language Models for News Summarization*. 2023. arXiv: 2301.13848 [cs.CL]. URL: https://arxiv.org/abs/2301.13848.

[77] Olaf Zimmermann. "Overcoming the research-practice gap: Root cause analysis and topics of practical relevance in software architecture and distributed systems." In: *Journal of Systems and Software* 230 (2025), p. 112539. ISSN: 0164-1212. DOI: https://doi.org/10.1016/j.jss.2025.112539. URL: https://www.sciencedirect.com/science/article/pii/S0164121225002080.

[78] jdepoix. *YouTube Transcript API*. Website. Available online at https://pypi.org/project/youtube-transcript-api; visited on July 20th, 2026. 2026.