Emerging Tech Trends: A Time-Series Topic Clustering Framework for Early-Stage Venture Capital Decision Making

Philip Tham and Juanita Santofimio

May 2025

Table of Contents

Abstract

This project aims to apply machine learning techniques to help tech investors in the venture capital (VC) industry identify emerging topics that could become “the next big thing.” To do so, we designed an algorithm to detect tech topics that start small but grow over time using time-series topic analysis. Our data consisted of text scraped from the tech-focused news site Hacker News over seven weeks. We extracted topic representations with BERTopic, tracked them through time via K-means clustering, and scored them based on investor relevance. We find that the model successfully identified dominant themes during this period—such as “Political Economy” and “Web Browsers”—but further tuning and a longer time window would be necessary to surface more specific topics with genuine investment potential.

Data Scraping README:

HackerNews Scraper for Articles and Comments
This data scraper retrieves article titles and comments from the HackerNews website for a given date range, and returns them as a text file in which each line corresponds to one article.

Dependencies: `bs4, requests,` optionally, `fake_useragent`
Parameters: date range, pages per day.

Usage:
In the last cell of the notebook:

1. Update the day1 and day2 params which establish beginning and ending of date range to fetch. However, multiday requests often fail, so it's better to make both days consecutive, hence why there is no param for month.
2. Update desired output file name.
3. Update the desired page count to fetch per day.
4. Run cell.

Output:
Text file where each row has one article title immediately followed by its comments on the hackernews site.

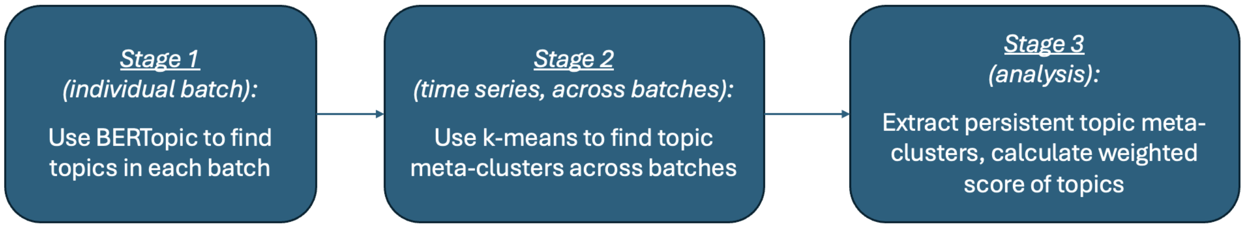
Problem specification

In venture capital (VC), investors derive their returns from identifying early stage companies with outstanding growth potential. To do so, their deal sourcing must satisfy multiple criteria, including:

1. Respond to industry painpoints: identify issues that the tech world can step in to address.
2. First mover advantage: identify opportunities before others in the same industry to minimize the cost of investment.
3. Risk mitigation: identify opportunities based on their capacity to grow and develop over time, rather than responding to momentary "hype".

An early-trend detection analysis could inform these criteria. Such an analysis can be carried out with applied machine learning through a time series topic analysis. In this project, **we attempt to design an algorithm that identifies tech topics that start small but grow over time**. These persistent and growing topics could represent investment opportunities.

This is implemented in a three stage approach:


Stage 1
(individual batch):
Use BERTopic to find topics in each batch
Stage 2
(time series, across batches):
Use k-means to find topic meta-clusters across batches
Stage 3
(analysis):
Extract persistent topic meta-clusters, calculate weighted score of topics

Data overview

Data was scraped from the article‑sharing website Hacker News. As a specialized, moderated platform that brings together participants from across the technology industry, it contains relevant insights for tech-oriented venture capitalists. Its publishing guidelines ensure higher quality and relevance of data, effectively acting as a first barrier against noise.

Scraping was done with a custom API to retreive the article titles and comments of the top ~90 articles for every day, which are ranked by the HackerNews algorithm based on freshness and popularity. We scraped data from March 1, 2025 to April 12, 2025, resulting in a total of 43 days and 3909 articles data. A single document consisted of the article's title and comments. Documents were grouped into batches of seven sequential days. There was a stride of three days between each batch.

This data is big, complex, and ripe for the application of machine learning algorithms. Assuming each article has 1000 words and the average reading rate is 250 words per minute, 3909 articles would take 260 hours (over 10 days). Venture capitalists operate on a tight schedule and do not have time for reading articles. Even if they do, the sheer volume of articles means that they probably would not be able to remember and classify each document correctly, and they also would not be able to come up with an accurate representation of each topic or topic meta-cluster after going through all the documents. Machine learning clustering would be optimal in dealing with such big data.

We conducted an initial **exploratory data analysis (EDA)** to determine the quality of information in each document. Each document had an average of 959 unique non stop-word tokens, with the stop-words being based on CountVectorizer's stop-word list. To test out how much information about an article was captured by the title and comments (our definition of a document), we calculated a cosine similarity between the embeddings of the document representation and the actual full-length document. For a toy example of the first document of the first batch, "The world's most unhinged video wall (made out of Chromebooks)," the similarity score was 0.91. This means that our document representations made out of the title and the comments represent the actual article's contents pretty accurately.

Stage 1: Generating topic representations from each batch

In the first stage, we generated **topic representations for each batch**. First, we used an optimized BERTopic topic modelling algorithm to identify topics. BERTopic does this by creating representations of each document, reducing the dimensionality of these embeddings, clustering the embeddings, and then generating a representation of each topic (i.e. the cluster of embeddings). The topic representations are a selection of the top n words in each topic's vocabulary with the highest TF*IDF scores. Second, we use an off-the-shelf embedding model to generate 768-dimensional embeddings for each topic representation.

Using BERTopic off-the-shelf yielded bad results as the model often found small clusters, identified too many outliers, and created topic representations that included many stopwords Several hyperparameters were thus tuned to optimize the BERTopic configuration.

Hyperparameter	How was this tuned?	Optimum value
Embedding model to tokenize and embed documents	Trial and error with other embedding models	hkunlp/instructor-base
Number of components for reduced document embeddings	Plotting out variance explained as number of components increase	40—this is a high number of dimensions but the document embeddings were of 768 dimensions and below 40 principal components yielded variance explained of less than 0.5
Minimum cluster size for HDBscan	Created calc_semantic_sim_with_penalty() objective function that measures the semantic similarity of points in a cluster while penalizing small clusters and the proportion of unclustered noise points (see aml_bertopic_opt_scratch.ipynb)	5
Number of words in each topic representation	Trial and error with [5, 10, 15, 20, 25, 30] and eyeballing the semantic similarity of documents within each cluster	20
Minimum number of topics for BERTopic model to find	Trial and error based on running BERTopic model algorithm 5 times and getting the mean number of topics created	5

The 20 word topic representations were then embedded using the same embedding model (hkunlp/instructor-base) to generate a 768-dimensional vector for each topic in the batch.

This process of using BERTopic to identify topics and then generating topic embeddings was repeated across all 13 batches of data, generating a total of 79 topics, with each batch having an average of 6 topics. Of all 3909 documents, 1377 were successfully categorized into topics (35%) by the HDBscan algorithm.

Stage 2: Time series analysis of topic meta-clusters across batches

From the previous stage, we generated one 768-dimensional embedding for each of the 79 topics across 13 batches. Now in stage 2, we want to do a time series analysis to look for meta-clusters of topics across time. First, we used PCA to generate lower-dimensionality embeddings for each topic. Second, we conducted K-means clustering on the 79 embeddings to find meta-clusters of topics across the batches. We chose to use K-means clustering instead of HDBscan (which was used under the hood in BERTopic) because HDBscan has a tendency to drop points as noise. We did not want the clustering in this stage to do this since many documents were already dropped in the topic identification stage, even with the minimum cluster size hyperparameter of HDBscan tuned. We thought that noise documents had already been filtered out in the previous stage, and that we should not be classifying whole topics as noise points in this stage.

The hyperparameters tuned for this stage to optimize the performance of the PCA and K-means algorithms are detailed below.

Hyperparameter	How was this tuned?	Optimum value
Number of components for reduced topic representations	Plotting out variance explained as number of components increase	10—variance explained at 10 components is above 0.5. We also wanted to keep this dimensionality low so that the K-means clustering works well.
Number of clusters for K-means	Plotting the ratio of within-cluster dispersion to between-cluster dispersion for 2 to 19 clusters	10

The distribution of meta-topics across each batch is shown in the stacked bar chart below. The top 20 most frequent words across all topic representations in each meta-cluster were also found to get a sense of what each meta-cluster meant (included in the Github repository under meta_topics_df.csv). Some key points that will be relevant for our analysis later are:

- Not all meta-clusters are persistent. For example, meta-cluster 8 (memory-hardware-cpu-apple-xbox) is only present in batches 1, 3, 11, 12, and 13. Another example is meta-cluster 0 (trump-political-https-violent-twitter). Other meta-clusters are persistent across batches like meta-cluster 1 (think-really-don-ai-just, probably about LLMs and AI “thinking”) or 4 (rust-language-code-type-memory).
- The size of meta-clusters vary across time. For instance, meta-cluster 6 (trump-government-money-market-tesla) starts off small in the first few batches, but gradually increases in size towards the last few batches.

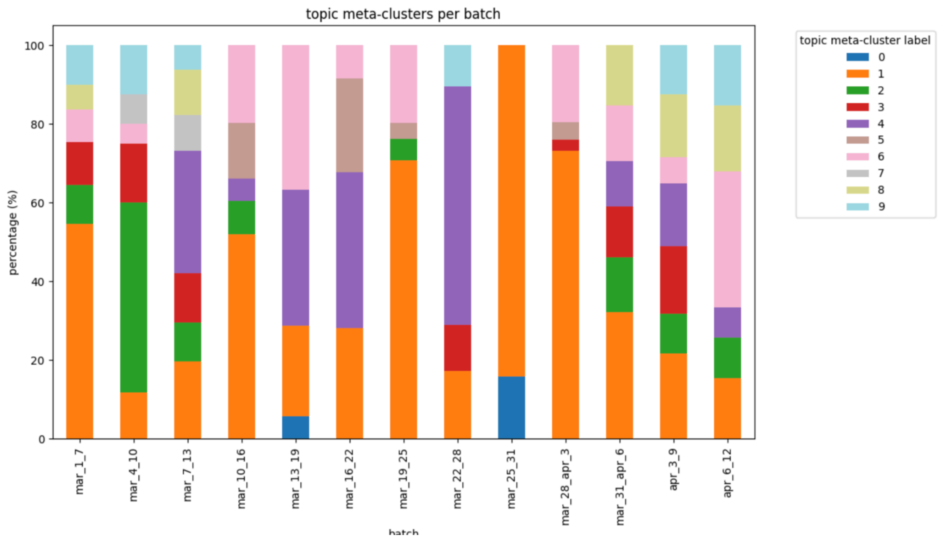


Figure 1: Cluster size by document count over time

Representations of meta-topic clusters clustered by K-means

Cluster Number	**Number of topics**	**Number of Docs**	**Meta Topic Representation**
0	3	28	['trump' 'political' 'https' 'violent' 'twitter' 'think' 'security' 'social' 'privacy' 'power' 'politicians' 'vote' 'platform' 'phishing' 'passwords' 'pen' 'microsoft' 'media' 'mail' 'mx']
1	20	603	['think' 'really' 'don' 'ai' 'just' 'things' 'work' 'code' 'people' 'make' 'llms' 'language' 'llm' 'like' 'software' 'company' 'doesn' 'reasoning' 'models' 'human']
2	8	124	['google' 'browser' 'firefox' 'https' 'chrome' 'mozilla' 'web' 'chromium' 'browsers' 'ads' 'safari' 'don' 'privacy' 'really' 'apple' 'just' 'ublock' 'ad' 'brave' 'money']
3	7	83	['war' 'trump' 'russia' 'europe' 'ukraine' 'nato' 'eu' 'military' 'countries' 'just' 'nuclear' 'putin' 'weapons' 'russian' 'country' 'european' 'peace' 'china' 'american' 'nukes']
4	10	174	['rust' 'language' 'code' 'type' 'memory' 'languages' 'compiler' 'really' 'don' 'python' 'unsafe' 'write' 'zig' 'venv' 'project' 'pyenv' 'packages' 'pip' 'macros' 'just']
5	4	45	['country' 'immigration' 'border' 'trump' 'visa' 'violence' 'government' 'canada' 'https' 'risk' 'police' 'surveillance' 'way' 'travel' 'phone' 'privacy' 'private' 'panopticon' 'law' 'public']

6	12	178	['trump' 'government' 'money' 'market' 'tesla' 'energy' 'think' 'tax' 'taxes' 'economic' 'don' 'currency' 'cars' 'economy' 'irs' 'federal' 'gold' 'musk' 'power' 'like']
7	2	19	['visa' 'terrorist' 'rights' 'support' 'law' 'protests' 'court' 'deported' 'israel' 'amendment' 'green' 'hamas' 'speech' 'trump' 'immigration' 'khalil' 'news' 'legal' 'political' 'freedom']
8	7	59	['memory' 'hardware' 'cpu' 'apple' 'xbox' 'windows' 'security' 'performance' 'model' 'gpu' 'firmware' 'software' 'ram' 'bluetooth' 'engines' 'os' 'intel' 'engine' 'esp32' 'devices']
9	6	64	['think' 'cancer' 'coffee' 'blood' 'sugar' 'study' 'history' 'health' 'immune' 'food' 'ireland' 'just' 'paternity' 'research' 'famine' 'aspirin' 'child' 'risk' 'placebo' 'skin']

As a sanity-check to verify the validity of the K-means clustering, we also plot the first two (of ten) dimensions of each topic and color-coded the points based on the topic meta-cluster. An analysis of just the first two dimensions shows the K-means algorithm forming promising clusters of topics that are close to each other.

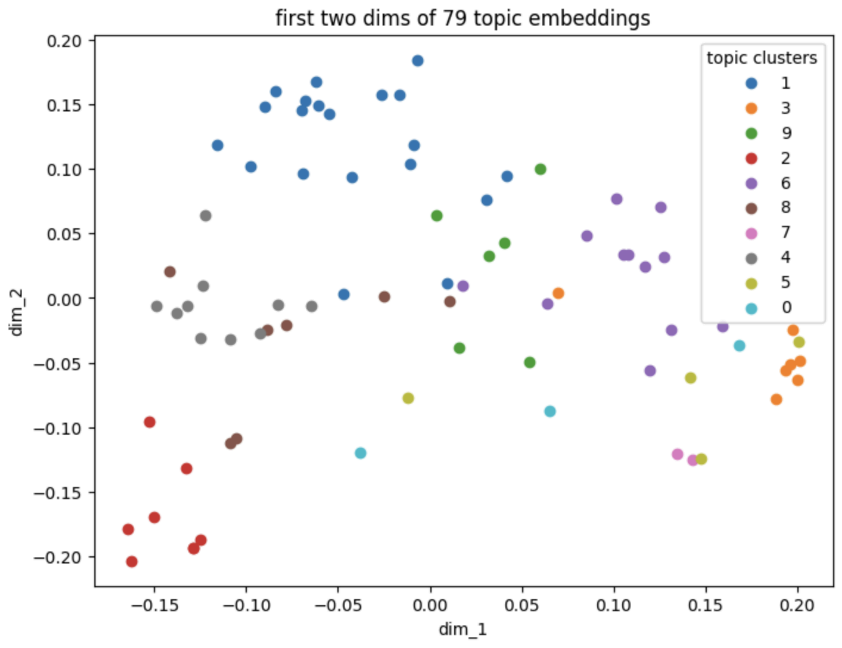


Figure 2: Scatter plot of the first two dimensions of the each topic meta-cluster

Another sanity check we did was to "cross validate" and check the stability of the clusters predicted by K-means. We chose the first 30 (of 79) topic representations, reduced the embeddings to 10 dimensions using PCA, and then used K-means to find 10 clusters. The semantic representation of the ten meta-clusters found are somewhat similar to (although obviously not exactly the same as) those found in the actual analysis involving all 79 topics. This is reassuring as it means that the algorithm provides somewhat stable results.

Clusters after PCA and K-means:

Cluster Label	Number of Topics	Number of Docs	Meta-Topic Representations
0	3	42	['trump' 'money' 'government' 'years' 'taxes' 'tax' 'tariffs' 'stock' 'spending' 'social' 'think' 'tesla' 'president' 'prices' 'ponzi' 'power' 'market' 'like' 'irs' 'federal']
1	4	89	['mozilla' 'google' 'chrome' 'browser' 'https' 'web' 'don' 'firefox' 'browsers' 'chromium' 'ads' 'really' 'ublock' 'just' 'apple' 'ad' 'librewolf' 'money' 'privacy' 'way']
2	3	34	['visa' 'law' 'green' 'rights' 'israel' 'protests' 'support' 'immigration' 'court' 'hamas' 'amendment' 'deported' 'terrorist' 'way' 'speech' 'khalil' 'political' 'news' 'legal' 'trump']
3	7	209	['code' 'memory' 'just' 'language' 'things' 'ai' 'rust' 'really' 'don' 'hardware' 'apple' 'compiler' 'languages' 'think' 'vram' 'write' 'typescript' 'uefi' 'understand' 'undocumented']
4	3	44	['war' 'ukraine' 'trump' 'nato' 'military' 'russia' 'europe' 'countries' 'weapons' 'russian' 'nuclear' 'eu' 'just' 'putin' 'country' 'tariffs' 'power' 'peace' 'nukes' 'european']
5	2	18	['paternity' 'immune' 'transgender' 'tattoos' 'tattoo' 'vaccine' 'sugar' 'study' 'skin' 'studies' 'risk' 'research' 'patients' 'transgenic' 'outbreak' 'measles' 'hormones' 'health' 'father' 'donation']
6	3	47	['work' 'company' 'really' 'make' 'engineers' 'software' 'think' 'engineer' 'don' 'ai' 'time' 'tech' 'things' 'working' 'money' 'llm' 'llms' 'management' 'managers' 'openai']
7	1	6	['venv' 'uv' 'tqdm' 'tools' 'tool' 'sync' 'pytorch' 'python' 'pyenv' 'projects' 'project' 'pip' 'pandas' 'packages' 'make' 'github' 'env' 'dependencies' 'conda' 'code']
8	3	31	['trump' 'energy' 'violent' 'think' 'social' 'ribosome' 'twitter' 'regulations' 'reconsideration' 'political' 'process' 'plants' 'oil' 'media' 'platform' 'market' 'make' 'hydrogen' 'just' 'https']
9	1	15	['world' 'time' 'think' 'make' 'like' 'just' 'ireland' 'history' 'food' 'famine' 'discworld' 'cancer' 'blood']

Stage 3: Finding persistent, small, and growing topic meta-clusters

The goal of the third stage was to pinpoint the persistent clusters that were small, but growing overtime. This was motivated by the necessity of venture capitalists to determine if a topic is small enough to be a new investment oportunity, and if the topic relevance is short-lived or if it gains traction over time.

To identify persistent clusters, batches were divided into thirds. Then, clusters with at least one batch in each of the three thirds were considered persistent. This resulted in selecting seven out of the nine clusters that had initially emerged in stage two. All these clusters share that they have at least a biweekly recurrence within the seven week period studied.

Persistent clusters with high scoring clusters highlighted (see score definition below):

Cluster label	**topic_rep**	**Cluster Name**
0	[twitter media bsky social bluesky political trump violent dictator clients https what is platform other just but beef are done]	Social Media Platforms & Political Discourse
1	[ai language memory think just other some don rust me be is get code have you do things only really]	Ariticial Intelligence
2	[mozilla firefox browsers browser chrome chromium privacy https brave ads web but google other way really don librewolf much now]	Web Browser and Online Privacy
3	[ukraine putin nato russia war russian nukes weapons we europe us trump our nuclear military peace not just what much]	Russia-Ukraine War & Nuclear Security
4	[compiler typescript runtime languages language rust code ts type javascript be things js like about better get than any to]	Programming Languages & Development Tooling
5	[immigration citizenship citizen visa canadian border canada law facebook what green about us has country who government way being not]	Immigration Policy
6	[congress federal government spending president money taxes irs are which we us trump what tax the who drones being is]	Political Economy

9	[famine aspirin cancer coffee food what about think being than irish not mice as child very history re ireland only]	Biomedical & Public Health Research

Growth was measured as the average increase in document count from batch to bactch. All seven persistent topics had positive growth on average. The growth rate was transformed into a size-growth score by dividing it by the cluster size. The purpose of this was to identify small clusters—*niche topics*—using large document counts as a penalty. By this measure, the most relevant topics were "Political Economy," "Web Browser and Online Privacy," followed by "Programming Languages & Development Tooling."

Cluster Label	**Average Growth Rate**	**Count**	**Score (Avg growth Rate / Count)**
6	0.737181	178	0.004141
2	0.476692	124	0.003844
4	0.512992	174	0.002948
1	1.579051	603	0.002619
3	0.162698	83	0.001960
9	0.087814	64	0.001372
5	0.009150	45	0.000203

Evaluating model performance

We compared the performance of our time series topic clustering algorithm to a dumb and reasonable model—using TF*IDF to get the top 20 non-stop words from each batch (the 13 csv files are included in the Github repository as top_20_tfidf_all_batches.zip), and then asking ChatGPT to find clusters of the words. This method found ten clusters across the 13 batches, the natural language representations of which were generated by ChatGPT 4-o:

Category label	**Examples**	**Description**
Scientific and niche technicals topics	Adhd, bioinformatics, harfbuzz, flatpal, kerala, wasm, swift	This cluster includes biomedical, linguistic, and low-level system software terms. It reflects deep technical domains (like bioinformatics, typography engines, or regional languages) that often have academic or niche software relevance.
Programming languages and web frameworks	Django, furhark, scala, svelte, phpstan, jepa, ambermoon	This group gathers programming languages, compilers, and tools used in modern software development. Some are newer or experimental (jepa, futhark, phpstan), while others are web or app focused.
Organizational tools and infrastructure	Liberapay, ory, rdp, overload, contention, frenzy	This cluster contains project management, identity, remote access, and productivity tools, often used in software infrastructure or team environments.
Lightweight DevOps/Data tooling	Duckdb, flash, fuchsia, terraform, git, dnssec	Focused on modern data systems and developer utilities, this group includes lightweight databases, deployment/config tools, and version control technologies.
Media, UI, and Sensory-Driven Experience	Cocaine, compress, discord, ipad, lonely, maestro, mains	A mix of software used for communication, media, or UX (e.g., discord, ipad, compress) with some emotional or performative elements (maestro, lonely, cocaine), perhaps reflecting a creative or expressive computing domain.
Compiler Toolchains and Data Processing	Gleam, ontology, magnesium, polars, css, redis	A mix of frontend frameworks (CSS, Gleam), high-performance data tools (Redis, Polars), and structured data semantics (ontology). Likely represents developer work at the interface between programming languages, data pipelines, and frontends.

Hardware Integration and Physical Computing	Lisp, mcp, prolog, zeromq, uv, trouble	This cluster is closer to embedded systems, hardware protocols, and low-level programming paradigms. Languages like Lisp and Prolog, communication layers like ZeroMQ, and hardware control markers like MCP hint at robotics or electronics integration.
AI Research, Experimental Tech and Automation	Jepa, oracle, ontology, alignment, trap, rl	A research-heavy grouping including AI alignment, reinforcement learning (rl), and knowledge-based systems (ontology, oracle). These tokens point toward experimental, philosophical, or safety-focused AI research themes.
Audio, Signal Processing, and Edge Use-Cases	Bass, ammonia, passion, sound, tcl, trap	A more auditory and sensor-rich category, containing keywords suggestive of acoustics or physical simulation (bass, ammonia), and lightweight scripting or edge tech (tcl). Could relate to audio processing or embedded ML.
Social and Infrastructure Software	Discord, tailscale, ruby, fortran, docker, depot	Combines community-facing tools (discord, tailscale) with common runtime environments (ruby, docker) and scientific computing tools (lfortran). This reflects full-stack engineering with social and scientific touchpoints.

This approach makes sense and finds some clusters that are similar to the ones found using our model, such as the cluster on hardware, development tools, and browsers. However, this dumb and reasonable model has several limitations:

- Using the top n TF-IDF tokens in each batch as a proxy for topics can overweight idiosyncratic or short-lived terms, miss multi-word expressions, and fail to capture semantic relationships, leading to fragmented or noisy clusters that may not align well with real conceptual boundaries.

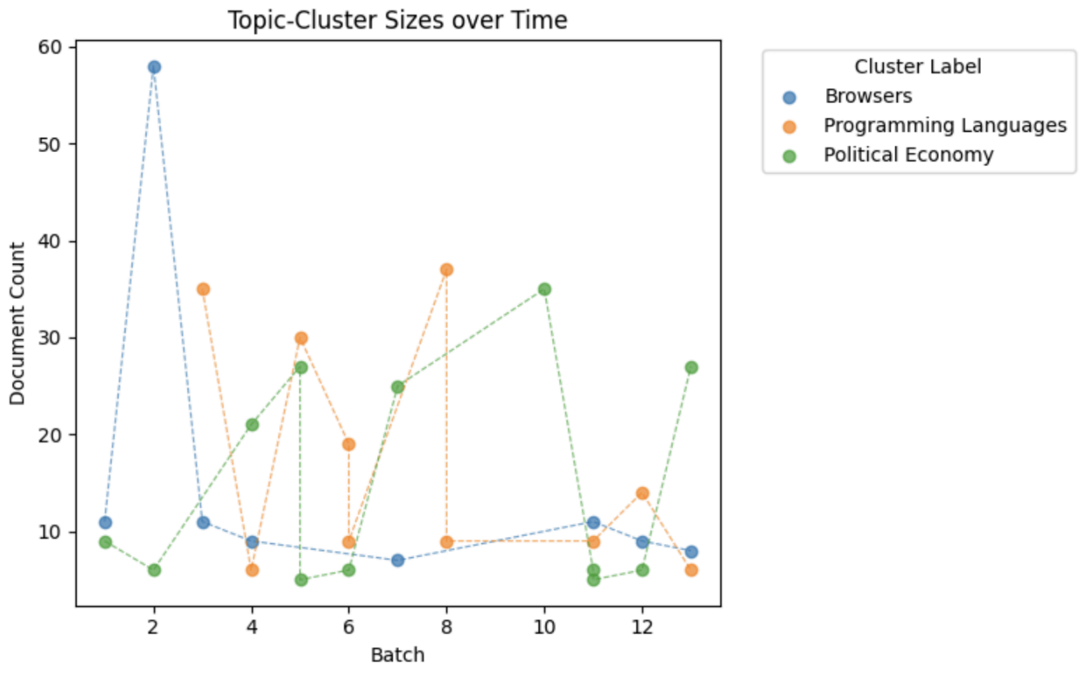
- The model only picked up tech topics, completely neglecting non-tech topics like articles on political economy (Trump's tariffs, trade wars etc) even though a cursory glance at HackerNews reveals many articles talking about these topics.

- Some words were clustered into multiple different categories, which could indicate hallucination by the LLM.

Because of the omission of non-technical news, which we know from eyeballing the data set on HackerNews, and the clustering of the same words into multiple different categories, we are fairly confident that our model outperforms this dumb and reasonable approach on accuracy.

Topic by Topic Analysis

Based on the model output, the topics that gained momentum during March 2025 were: "Political Economy," "Web Browsers and Online Privacy," followed by "Programming Languages & Development Tooling."



Top 3 Scoring Topic-Document count across batches

Political Economy:

The high growth score for the Political Economy category was not surprising given that it was a major topic across all news sectors, including the tech space. Although not perfectly, it was the topic that most closely resembled an increasing document count trend. Therefore, while it is not the type of topic that would be informative for a tech investor in terms of identifying industries for deal sourcing, it demonstrates that the algorithm captures a strong signal from the current tariff landsape being significant to the industry. In other words, this topic exhibitted the type of behavior that the model would flag as relevant under the size and growth assumptions.

Web Browser and Online Privacy:

The second place ranking for the Web Browser and Online Privacy was initially promising as it identifies a specific issue in the tech space, as well as pinpointing specific products like "Chrome" and "Mozilla." The persistence of this topic throughout time could represent an investment opportunity for a VC firm in startups that address privacy concerns in web browsing. However, upon further inspection, it is evident that the high score of this topic is mainly driven by one big jump betwen the first two batches, which coincides with the Mozilla announcement of rolling out updates to fix a critical security flaw in Firefox for Windows, just days after Google patched a similar zero-day in Chrome. This highlights that the scoring model does not accurately reflects the start small and grow trend that it aimed to capture. Verifying that the the count is monotonically increasing would more accurately reflect this behavior.

Programming Languages & Development Tooling:

The third topic also pointed to a more specific subsector of the tech industry. In this case, the high growth-size score was mainly driven by multiple jumps from low to high document in the first half of the timeline. This again highlights that the score mechanism is biased towards intra period growth signals, rather than an overall growth trend. Nonetheless, this topic exhibits other desirable topics for a VC analyst. First, it captures the novelty effect that the model is aiming to uncover, as it is bringing forth programming languages like Rust, which is gaining popularity thanks to its enforcement of memory safety at compile time. As an investor, this is an opportunity to build recurring revenue because startups that provide tools for emerging coding spaces have the potential to lock in clients for long periods of time, and can eventually be bought by bigger companies. Even with the risks of a low adoption rate of the new languages and tools, the idea of looking into how new products address the painpoints of developers is a worthwhile insight derived from the model.

Limitations and further extension

The performance of our algorithm could be optimized even further if several parts of the system were refined.

- **Stride length for batches**. In the data collection and preprocessing step, we defined each batch as the top ~90 articles on HackerNews across a week. Each batch was 3 days off from the following batch (ie, if batch 1 covered March 1 to 7, batch 2 would cover Mar 4 to 10). We did this to maximize the amount of data that our model could train on. However, having this short stride length introduces overlap between batches that can lead to data redundancy, decreasing the marginal gain in unique information. Batch stride length could be increased to minimize overlap going forward.

- **Minimum number of topics per batch for BERTopic topic identification**. In stage 1, the number of topics generated by BERTopic sometimes varies across runs, even after we fixed several hyperparameters for the PCA and HDBscan that run under the hood of the model. We arbitrarily set a minimum of five topics for BERTopic to find in each batch. This was loosely based on eyeballing the number of topics that the model generated across several runs and batches. More quantitative and systematic tuning could be done to figure out what a more accurate minimum number of topics could be.

- **Threshold for considering a topic meta-cluster "persistent."** In stage 3 of our algorithm, we split the 13 batches into thirds and considered a meta-cluster persistent if topics from the meta-cluster existed in each third. A more precise method of measuring the consistency of topic meta-clusters could be thought of for a more accurate analysis.

- **Final score calculation—gradient measurement**. In stage 3, we calculated a final score that measured the growth of a topic meta-cluster across batches. This was done by taking the average of the gradients at each batch. However, we realized that this method of averaging out the gradient was not foolproof. For instance, for topic meta-cluster 2 on browsers, we realized that the averaged gradient was inflated because of a large increase in document count between batches 1 and 2, while the gradient was relatively small across other batches (see the line graph in the "Analysis" section). An alternative measure of the growth of a topic meta-cluster could be refined to account for such cases.

- **Final score calculation—weights for meta-cluster size**. We weighted the final score by the size of the meta-cluster (ie, the total number of documents across all batches that fell under this meta-cluster) to penalize large meta-clusters. We added this penalty because, as VC investors, we want to look out for small meta-clusters that gradually grow in size and not large ones that are already obvious. We included an alpha hyperparameter in our algorithm that determines the size of this size penalty and arbitrarily set this to 1 in this analysis. Future analysis could tune this hyperparameter, or come up with a more robust method of penalizing large meta-clusters (or meta-clusters that start out large in the beginning) that would not be of much use to VC investors looking for the "next big thing."

Once the hyperparameters of this pipeline are optimized, data from a longer time period (we only scraped a month and a half of data in this project) can be extracted and fed into the model. From there, an extension to the pipeline that we have developed here would be a predictive algorithm that models how likely a particular topic meta-cluster would grow into a persistent and large meta-cluster across batches. In other words, such a system would be able to predict whether a topic is the “next big thing” or not. Developing such a model would require labelled data of which meta-clusters are “next big things” for supervised learning, and would also require tuning the alpha hyperparameter in the final score calculation weights.

Another extension would be to expand the application of this model to specialized news outlets within the tech sector like biotech, or fintech, among others to improve the level of specificity of the output and better identify profit opportunities.