Scientific Concept Evolution Tracker

Nic Bolton
University of Toronto
Toronto, Canada
nic@cs.toronto.edu

Abstract

With the sheer volume of research being published, the history and context of how scientific ideas evolve are often difficult to visualize through the noise. Terminology in science can be dynamic—the semantic meaning of terms such as "neural networks", "entropy", or "plasma" shift significantly over decades as new research subfields emerge. Traditional information retrieval systems and static vector databases index semantic meanings as fixed points in high-dimensional space, ignoring time as a dimension which hides the evolutionary history of these concepts. This report introduces the Scientific Concept Evolution Tracker (SCET), a comprehensive system designed to ingest, index, and analyze large-scale scientific corpora to quantify this semantic drift. SCET is built for scale with PostgreSQL for storing metadata and Milvus for embeddings. We introduce a methodology that combines unsupervised clustering (K-Means) with temporal segmentation (Decision Tree Regression) to automatically identify distinct "eras" of a concept's life cycle. We demonstrate the system's capabilities through case studies, such as the divergence of "Transformer" from electrical engineering to natural language processing, and provide a quantitative analysis of system performance on a dataset sourced from arXiv.

1 Introduction

Science is a cumulative process, yet the language of science is fluid. A core challenge in bibliometrics and the emerging field of "Science of Science" (SciSci) is understanding how scientific consensus and terminologies evolve. SciSci seeks to use data-driven methods to understand the mechanisms of scientific discovery, yet it often struggles with the large scale and unstructured nature of the literature. For a researcher entering a new field, understanding the historical context of a term is as critical as understanding its current definition. For example, a query for "Attention" in 2005 would yield results dominated by cognitive psychology and neurobiology. The same query in the late 2010s and early 2020s is overwhelmingly dominated by field of machine learning. This is the phenomenon that we are interested in, that is, being able to quantify how a concept's relevance or association with specific fields change over time. This problem is compounded by two different factors, specifically, *polysemy* (one word having multiple meanings, like "Attention") and *synonymy* (multiple words having the same meaning, like "Deep Learning" and "Hierarchical Feature Learning"). Traditional keyword-based systems struggle with both, often retrieving irrelevant documents or missing relevant ones.

The introduction and combination of Large Language Models (LLMs) and vector databases have revolutionized semantic search. By representing text as dense vectors in a high-dimensional space, we can capture semantic similarity beyond simple keyword matching. However, most vector search implementations treat the document corpus as a static snapshot. They are designed to answer the question, "What is semantically similar to this query now?" rather than "How has the meaning of this query changed over time?".

This project addresses the following question. How can we design a scalable vector database system that can quantify the semantic evolution of scientific concepts and identify pivotal publications that drive these shifts?

SCET was developed to answer this question. It consists of a pipeline that:

(1) Ingests and indexes scientific abstracts using a hybrid embedding strategy
(2) Retrieves context-aware results using a weighted hybrid search
(3) Clusters results into sub-concepts associated with a given query
(4) Produces a set of time periods that represent "stable" eras of a sub-concepts association/relevancy
(5) Identifies papers that are "pivotal" to the shift into these eras

The remainder of this paper is organized as follows. We first review the background and related work in fields relevant to SCET. We then discuss the details of the methodology and system architecture, followed by a presentation of the experimental results and case studies. We conclude by discussing limitations and future work.

2 Background and Related Work

The problem to solve (tracking the evolution of scientific concepts) sits at the intersection of Natural Language Processing (NLP), Information Retrieval (IR), and the Science of Science. This section reviews the historical progression of these fields and the specific technologies that enable SCET.

2.1 The Evolution of Information Retrieval

The field of IR has evolved through several distinct paradigms. An overview of the history is listed below. It is important to note that although these systems perform well within their own goals, they all lack the ability to capture semantic and contextual meaning within terms.

2.1.1 Boolean Logic. The earliest IR systems relied on semantics used within set theory, where queries were built with operators such as AND, OR, NOT. These systems could make retrievals at a high level of precision by using *inverted indexes.* These work as a key-value store, that is, each term is a key that is associated with a list of pages that contain the term. This results in a very fast and precise system [26].

2.1.2 TF-IDF. Term Frequency Inverse Document Frequency (TF-IDF) introduced the concept of weighting the importance of words. This system involves calculating a score for a term, derived by balancing how frequently a term appears in a specific document (TF) against how rarely the term appears across the entire collection of documents (IDF). Consequently, these scores often favour the terms that best characterize the topics involved in a collection of documents [15].

2.1.3 Vector Space Model. The Vector Space Model (VSM) generalized the idea of weighting terms by representing documents as vectors in a multi-dimensional space, where each dimension corresponds to a distinct term in the corpus. In this model, the relevance of a document to a query is measured by the similarity between their respective vectors. The similarity measurement is often done using Cosine Similarity

$$\cos(\boldsymbol{q}, \boldsymbol{d}) = \frac{\boldsymbol{q} \cdot \boldsymbol{d}}{\|\boldsymbol{q}\|\|\boldsymbol{d}\|} \tag{1}$$

for TF-IDF weights of the query $\boldsymbol{q}$ and the document $\boldsymbol{d}$.

This allows for ranked retrieval results based on the angle between vectors, rather than a binary inclusion/exclusion. However, traditional VSMs treat terms as orthogonal dimensions, meaning they cannot capture semantic relationships between synonyms (for example, car and automobile) without explicit expansion [15].

2.1.4 BM25. Okapi Best Matching 25 (BM25) represents a significant evolution in probabilistic information retrieval. While sharing similarities with TF-IDF, BM25 introduces two critical improvements: term saturation and document length normalization. Unlike TF-IDF, where the score increases linearly with term frequency, BM25 applies a saturation function so that the value of a term diminishes as it is repeated (thus, preventing long repetitions of keywords from dominating results). Additionally, it penalizes long documents (which naturally contain more terms) to prevent them from unfairly dominating search results. BM25 remains a strong baseline for lexical search tasks today [25].

2.2 Evolution of NLP Representations

The representation of text is fundamental to our ability to measure drift. To quantify how a concept evolves, we must first map it to a mathematical space where its meaning is a measurable coordinate. The history of NLP can be viewed as a progression towards richer, more contextualized mappings. Early methods treated words as atomic symbols, making it impossible to measure the semantic distance between them. Modern approaches embed text into continuous vector spaces, allowing us to calculate the magnitude and direction of semantic shifts using geometric operations.

2.2.1 Static Embeddings. Word2Vec [18] and GloVe [21] introduced distributed representations. They rely on the distributional hypothesis: “a word is characterized by the company it keeps”. These models map discrete tokens to dense vectors in a continuous space, capturing syntactic and semantic regularities. The vectors that are assigned to these tokens (words) are derived so that they are in close proximity to other similar words (for example, the vectors for “dog” and “puppy” are very close). However, these models are static, meaning they assign a fixed vector to each word type regardless of context. For example, a vector would be generated for the word “bank”, but would be indifferent whether it was used in the context of river banks or piggy banks. This imposes a significant limitation for scientific text, where acronyms and terms often have distinct or field-specific definitions that cannot be derived via static representation.

2.2.2 Contextual Embeddings. ELMo [22] and BERT [6] introduced dynamic embeddings to address the limitations of static models. The Transformer [28] architecture’s self-attention mechanism allows the representation of a token to be a function of its surrounding context. BERT (Bidirectional Encoder Representations from Transformers) pre-trains on a masked language modeling objective, allowing it to learn deep bidirectional representations. Unlike static embeddings, BERT generates a representation for a token that is conditioned on the entire input sequence. This allows a word such as “bank” to become associated with the context it was used in—if it sees “river” nearby, then it can identify river bank as the likely association. This concept is useful for our problem, as it allows for distinguishing “Attention” (psychology) from “Attention” (machine learning) in a given corpus.

2.2.3 Domain-Specific Models. Although contextual embedding models such as BERT served as great drivers for advancing the field of NLP, they often underperform on domain-specific corpora such as scientific text. This is because they are trained on corpora such as Wikipedia, which differs significantly in vocabulary and syntax from scientific literature. SciBERT [3] addresses this by pre-training BERT on a large corpus of scientific papers. SPECTER [4] took this further by leveraging a unique feature of research: citations. They used citation graphs as a signal for semantic similarity, which groups papers based on how related they are as opposed to just textual overlap. It uses a triplet loss objective:

$$\mathcal{L} = \max\left\{d(q, p^+) - d(q, p^-) + m, 0\right\}$$

where d is a distance function, q is a query paper, p^+ is a cited paper, p^- is a non-cited paper (but may or may not be cited by p^+), and m is the loss margin hyperparameter. By training the model to pull cited papers closer in vector space and push unrelated papers apart, SPECTER learns embeddings that reflect the actual relationships between papers.

2.3 Vector Database Indexing

Searching a dataset of millions of high-dimensional vectors is computationally prohibitive using brute force ($O(N)$). This is where database indexes are necessary: they are created as a data structure that trade write latency and storage space for faster retrieval speeds. Without an index, the system would be forced to perform a full table scan, checking every single record to find a match. SCET utilizes Approximate Nearest Neighbor (ANN) algorithms to construct these indexes.

2.3.1 Inverted File Index. The Inverted File Index (IVF) works by partitioning the vector space into Voronoi cells, where every document vector is then assigned to its nearest centroid. Now upon search, the system can extract a subset of the vectors by comparing the query vector to the centroids. A benefit to IVF is its low memory footprint [15].

2.3.2 Hierarchical Navigable Small World. While partition-based methods like IVF offer memory efficiency, graph-based approaches currently provide the superior trade-off between latency and recall. Hierarchical Navigable Small World (HNSW) structures data into a multi-layered graph hierarchy inspired by Skip Lists and the "small world" phenomenon [30]. The upper layers consist of sparse, long-range links that allow the search algorithm to traverse the vector space rapidly, effectively zooming in on the target region. Once the coarse location is identified, the search descends to lower, denser layers for fine-grained greedy traversal to locate the nearest neighbors. The HNSW process is illustrated in Figure **??**. Although HNSW requires higher memory overhead to store the graph connectivity compared to quantization methods, it is robust against the curse of dimensionality and does not require the training phases of clustering approaches [14].

The performance of HNSW is governed by two key parameters: M, the maximum number of outgoing connections per node, and $efConstruction$, the size of the dynamic candidate list used during index construction. A higher M increases the connectivity of the graph, improving recall at the cost of memory and search time. A higher $efConstruction$ leads to a higher quality graph (better recall) but increases the indexing time. For SCET, we prioritize recall and query speed over memory footprint, making HNSW the ideal choice.

2.3.3 *Filtering Strategies in Vector Search.*

A critical aspect of vector database performance is how they handle metadata filtering (e.g., "find papers from 2015"). There are two primary approaches: *post-filtering* and *pre-filtering*. Post-filtering performs the vector search first to find the top k nearest neighbors, and then filters out the results that do not match the metadata criteria. This is simple to implement but can lead to disastrous recall if the metadata filter is selective (e.g., if the top k results are all from 2020, a query for 2015 will return zero results). Pre-filtering applies the metadata filter first to reduce the search space, and then performs vector search on the remaining subset. While this guarantees correct results, it can be inefficient if the index does not support filtered search natively. SCET utilizes Milvus's partition-key feature, which effectively acts as an optimized pre-filtering mechanism by physically segregating data based on the hash of the year and category.

2.4 Semantic Drift Analysis

Semantic drift (or semantic change) is the study of change with respect to the meaning of words, specifically the evolution of how a word is used. For example, the word *awful* originally meant to inspire wonder or fear, and hence impressive. Today, it is used to describe something that is regarded as very bad.

2.4.1 *Alignment.*

Since embedding models are initialized randomly, the vector spaces for different time periods often end up rotated or flipped relative to each other. Orthogonal Procrustes Analysis is used to fix this, by mathematically rotating the vector space of one time period to align it with the next. By locking the two maps together, it ensures that if a word's position changes, it represents a genuine shift in meaning rather than a side effect of the model's random initialization [16] .

2.4.2 *Dynamic Word Embeddings.*

While alignment-based methods rely on independent training of temporal slices, dynamic probabilistic models treat the evolution of semantic meaning as a continuous latent variable process. First introduced by Balmer and Mandt [2], their process involves training a single global model where the embedding of a term at time t is conditioned on its embedding at time $t-1$, modeled as a Gaussian Random Walk. This process ensures that meanings shift gradually rather than abruptly. By using data from surrounding time periods, these models work much better for rare terms. This eliminates the random noise often seen in year-by-year training, resulting in a clearer, smoother path of how a concept has changed.

3 Methodology

3.1 System Architecture and Data Pipeline

ArXiv has a public Google Cloud Storage Bucket that includes the metadata of approximately 4 million publications. For this project, we utilize a snapshotted version of this metadata, which is available on Kaggle [1] and is provided as a 3.8GB JSON file.

3.1.1 *Data Ingestion and Relational Storage.*

Due to the size of the dataset, loading the entire file into memory was infeasible. We resolved this by processing the file in batches. For each publication we extract its unique arXiv ID along with the title, abstract, primary category, publication date, and DOI. During this phase, we perform basic data cleaning: removing papers with empty abstracts, normalizing whitespace, and filtering out records with malformed dates. This ensures that the downstream embedding models receive clean, high-quality text input.

This structured metadata is stored in a PostgreSQL [10] database. PostgreSQL serves as the source of truth for document metadata and enables efficient filtering based on structured fields (for example, retrieving all papers from 2015 in the *cs.AI* category). The database schema is designed to handle the metadata efficiently, with the arXiv ID serving as the primary key that links the relational data in PostgreSQL to the embeddings in the vector database. This separation of concerns allows each system to be optimized for its specific access patterns.

3.1.2 *Vector Storage Layer (Milvus).*

We deploy Milvus Standalone v2.6.6 [29] as our vector database. Milvus was selected for its cloud-native architecture, which separates storage from compute, allowing us to scale query nodes independently of data volume. To optimize for our specific access patterns (which heavily rely on filtering by time and scientific domain) we implement a custom partitioning strategy. We define a partition key derived from a hash of the publication year and primary category. This allows the query engine to prune irrelevant partitions during search, significantly reducing the search space for temporal queries. For example, a query for *Machine Learning Papers in 2015* only needs to scan a small fraction of the total index, rather than the entire collection of 4 million vectors. The collection schema is detailed in Table 1.

We utilize Hierarchical Navigable Small World (HNSW) indexes for dense vectors. We configure the index with $M = 16$ (the number of bi-directional links per node) and $efConstruction = 200$ (the size of the dynamic candidate list during build). This configuration was selected to maximize recall while maintaining interactive query

Table 1: Milvus Collection Schema

Field Name	Data Type	Description
`paper_id`	Int64	Primary Key (64-bit hash of arXiv ID)
`arxiv_id`	VarChar(32)	ArXiv ID (for debugging)
`dense_vector`	FloatVector (768)	Semantic embedding (SPECTER2)
`sparse_vector`	SparseFloatVector	Lexical embedding (SPLADE)
`publication_year`	Int16	Scalar field for temporal filtering
`partition_key`	Int64	Hash of publication year and category

latency at the cost of higher memory consumption, compared to quantization-based indexes like Inverted File with Product Quantization [19]. For the sparse vectors, we use a Sparse Inverted Index. This is analogous to the inverted indexes used in traditional search engines, but optimized for floating-point weights.

3.1.3 Deployment and Infrastructure. The entire SCET system is containerized using Docker [17] to ensure reproducibility and ease of deployment. We utilize Docker Compose to orchestrate the necessary services, including Milvus Standalone as the core vector database engine, etcd [8] for metadata management and service discovery, MinIO [20] as an S3-compatible object storage service for persisting vector data, and PostgreSQL for storing structured paper metadata.

The system is deployed on an AWS EC2 instance (`m6gd.2xlarge`). The hardware consists of an arm64 CPU (8 vCPUs @ 2.50 GHz), 32GB of RAM, and 474GB SSD. Resource limits are enforced via Docker Compose, with Milvus allocated 20GB of RAM, etcd 2GB, and MinIO 4GB.

3.2 Hybrid Embedding Strategy

SCET employs a *Retrieve-then-Rerank* strategy using a hybrid score to balance semantic understanding with lexical precision. Pure lexical search often suffers from the *vocabulary mismatch problem* (or lexical gap), where it fails to retrieve relevant documents that use synonyms (for example, missing a paper on "automobiles" when querying for "cars") [9]. Conversely, pure dense retrieval can lack *lexical precision*, retrieving semantically related documents that miss specific query terms (for example, retrieving a paper about recurrent neural networks when the user specifically asked for LSTMs). We use a hybrid approach so that both of these issues can be addressed.

3.2.1 Dense Embedding (Semantic). We utilize SPECTER2 [27], which succeeds the discussed SPECTER model that is pre-trained on scientific citations. The input to the model is a concatenation of the paper's title and abstract, separated by SPECTER2's provided separator token: `Title[SEP]Abstract`. This generates a 768-dimensional dense vector that captures the high-level semantic intent of the paper. This explicitly encodes a "scientific relatedness" into the vector space: papers are encoded as queries/candidates that are intended for use in link predictions or nearest neighbor searches.

3.2.2 Sparse Embedding (Lexical). To prevent the loss of specific keyword information, we also generate sparse embeddings using SPLADE (Sparse Lexical and Expansion Model) [11]. SPLADE works by taking sentences or paragraphs as input and mapping them to a 30522-dimensional vector space. Unlike traditional bag-of-words models (such as TF-IDF) which only assigns weights to terms present in the document, SPLADE performs *term expansion*. It uses the BERT Masked Language Model to predict relevant terms that do not appear in the text but are semantically related, assigning them non-zero weights. Mathematically, the weight w_{ij} for token j in document i is calculated by aggregating the log-saturated attention weights across all input tokens. This functions as a "learned" inverted index, providing the precision of keyword matching with the recall of semantic expansion. This effectively allows us to get the benefits of both BM25 and dense retrieval [7].

3.2.3 Hybrid Scoring. Retrieval is performed by executing two parallel searches: an ANN search on the dense index along with a term-matching search on the sparse index. A common challenge in hybrid search is the *rank fusion problem*, that is, if we only retrieve the top k results from each index, a document that is relevant in both but not top-ranked in either might be discarded. To address this, we fetch a candidate pool significantly larger than the final limit (for example, $10 * k$) from both indexes. We then normalize the scores (since Cosine Similarity scores are bound to the interval $[-1, 1]$, and Inner Product is unbounded) and combine them using a weighted sum:

$$S_{hybrid} = \alpha \cdot S_{dense} + (1 - \alpha) \cdot S_{sparse} \tag{2}$$

where α is a hyperparameter controlling the trade-off between semantic and lexical relevance. We default to $\alpha = 0.5$, giving equal weight to conceptual similarity and keyword precision.

3.3 Concept Clustering Algorithm

To identify the distinct meanings (sub-concepts) associated with a query term, we perform clustering on the retrieved dense vectors. The underlying assumption is the Distributional Hypothesis [1] applied to vector space. Papers discussing the same sub-concept (for example, "Corona" as a virus) will form a dense cluster that is spatially separated from papers discussing a different sub-concept (for example, "Corona" as a solar feature).

Given a set of search results for a query Q, we extract their dense vectors V_Q. We then apply k-Means clustering to partition V_Q into k clusters. The optimal number of clusters k is determined dynamically for each query by maximizing the Silhouette Score over a range such as $[2, 8]$. The Silhouette Score s_i for a single data

[1] In linguistics and NLP, this states that words appearing in similar contexts tend to have similar meanings

point i is defined as:

$$s_i = \frac{b_i - a_i}{\max\{a_i, b_i\}} \tag{3}$$

where a_i is the mean distance between i and all other points in the same cluster (cohesion), and b_i is the minimum mean distance from i to all points in any other cluster (separation). The overall Silhouette Score for a clustering configuration is the mean s_i over all points. A score close to 1 indicates that the clusters are well-separated and dense, while a score near 0 indicates overlapping clusters. By maximizing this metric, SCET automatically determines whether a concept has split into 2 or more distinct sub-concepts without user intervention.

To interpret these clusters, we leverage the sparse vectors. Since SPLADE vectors represent the "lexical essence" of a document, we can generate a descriptive label for a cluster by summing the sparse vectors of all papers within it. The tokens with the highest summed weights represent the most discriminative keywords for that group. For example, one cluster for "Transformer" might yield top tokens "voltage, power, current", while another yields "language, sequence, attention". This serves as a convenient source for user interpretability.

3.4 Era Detection

Once sub-concepts are identified, we aim to detect their temporal "eras", that is, periods of stable activity or relevance. We treat the yearly publication count of a sub-concept as a time series signal $Y = \{y_t\}$. A naive approach might look for peaks or valleys, but publication data is noisy. Instead, we employ a Decision Tree Regressor to approximate this signal with a step function. The algorithm recursively partitions the time axis to minimize the Mean Squared Error (MSE) within each interval. For a given node representing time interval T, the algorithm searches for a split point t_{split} that minimizes the weighted impurity:

$$L(t_{split}) = \frac{N_L}{N} MSE_L + \frac{N_R}{N} MSE_R \tag{4}$$

where N_L (N_R) is the number of years in the left (right) partitions, and $MSE_{L,R}$ is the variance of the publication counts in those partitions. By fitting a decision tree with a limited depth (for example, a max depth of 3), the algorithm naturally finds the optimal split points that minimize the variance within each leaf. These split points (thresholds) correspond to the temporal boundaries where the trend significantly changes. The leaf nodes of the tree represent the stable eras (for example, low activity in 2000–2012 and high activity in 2013–2023). This method is non-parametric and robust to outliers, making it well-suited for bibliometric data.

3.5 Identifying Pivotal Papers

For each identified era, we seek to find the papers that best represent that specific semantic context during that time. Simply selecting the most cited papers would bias the results towards older, established work. Instead, we query the system for the top N papers within the specific time window of the era, ranked by their hybrid score relative to the original query. These papers serve as the "anchors" for understanding how the concept was defined and used during that period. By surfacing these pivotal papers, SCET provides a curated list of papers that guides the user through the history of the concept and highlights the seminal works that defined each era.

Table 2: Pipeline Latency Breakdown (N = 1000 Papers)

Pipeline Stage	Avg Time (s)	% of Total Time
Hybrid Search	1.25	31.34
Concept Clustering	2.73	68.48
Era Detection	0.00	0.07
Pivotal Paper ID	0.00	0.10
Total	3.98	100.00

Table 3: Clustering Scalability

Papers	Time (s)	Throughput (papers/s)
100	0.38	263.17
500	1.38	362.68
1000	2.82	354.66
2000	5.26	380.01
3000	9.81	305.70
4000	14.41	277.61
5000	17.02	293.71

4 Experimental Results

In this section, we evaluate the performance of SCET from two perspectives: system latency/scalability and retrieval quality. All experiments were conducted on the hardware described in the *Deployment and Infrastructure* section.

4.1 System Performance Benchmarks

To ensure the system is viable for interactive use, we benchmarked the end-to-end pipeline latency across a set of 10 distinct scientific queries. The pipeline parameters were fixed at $N = 1000$ papers retrieved, $\alpha = 0.5$, and automatic cluster detection.

4.1.1 End-to-End Pipeline Latency. Table 2 summarizes the execution time for each stage of the pipeline. The average total execution time for processing a query with 1000 retrieved documents was approximately 4 seconds. The results indicate that the *Clustering* phase is the bottleneck, consuming roughly 60-70% of the total compute time. This is expected, as K-Means clustering on high-dimensional vectors (768 dimensions) is computationally intensive. The *Search* phase (Milvus retrieval) averages around 1.25 seconds, which demonstrates the efficiency of the HNSW index. Notably, the *Era Detection* and *Pivotal Paper* identification steps are negligible, which confirms that the temporal analysis algorithms (Decision Tree Regression) are highly efficient once the data is structured.

4.1.2 Clustering Scalability. Since clustering is the dominant cost, we further analyzed its scalability by varying the number of retrieved papers from 100 to 5000. As shown in Table 3 and Figure 1, the system maintains a throughput of approximately 250-400 papers per second. While the time increases linearly with N, processing 5000 papers takes just above 17 seconds.

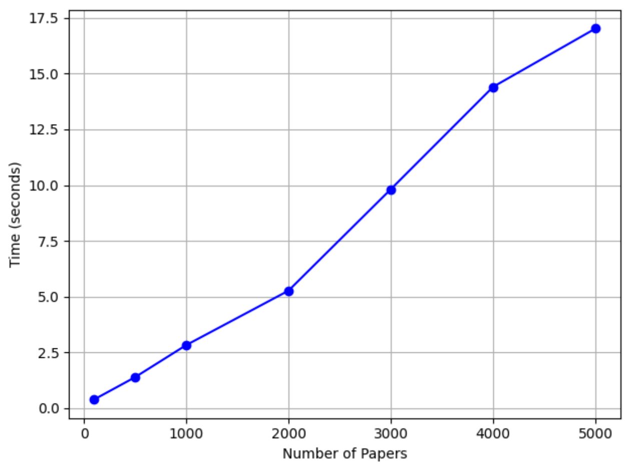

Figure 1: Clustering Scalability (Time vs N Papers)

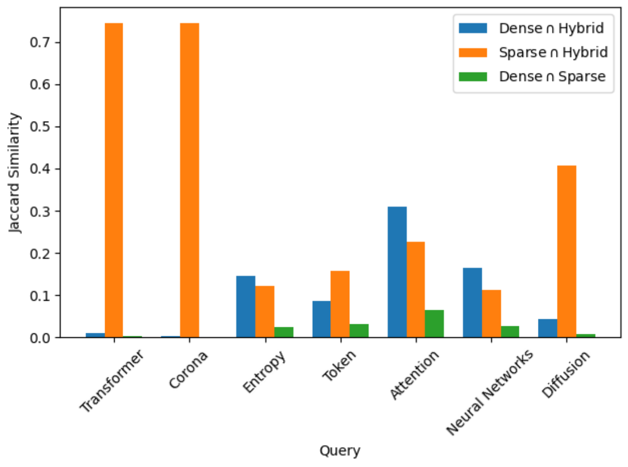

Figure 2: Overlap between Retrieval Methods (Top-50)

4.2 Ablation Study: Hybrid vs. Dense-Only

To validate the necessity of the hybrid embedding strategy, we compared the retrieval results of Dense-Only ($\alpha = 1.0$), Sparse-Only ($\alpha = 0.0$), and Hybrid ($\alpha = 0.5$) configurations. We measured the Jaccard Similarity between the sets of top-50 retrieved papers for five distinct queries.

The results in Table 4 and Figure 2 show a disjointedness between semantic and lexical search. For the query “Transformer”, the overlap between Dense and Sparse results was 0.0. This implies that the set of papers containing the exact keyword “Transformer” is completely different from the set of papers that are semantically close to the concept in the SPECTER2 embedding space. This perfectly illustrates the discussed vocabulary mismatch problem. The dense model likely retrieved papers discussing topics such as self-attention or NLP that are semantically identical but lack the specific keyword, while the sparse model strictly adhered to the token.

Interestingly, the Hybrid results ($\alpha = 0.5$) showed a high overlap with the Sparse results (75% for “Transformer” and “Corona”), but a more balanced mix for “Attention” and “Neural Networks”. This suggests that for highly specific technical terms, the sparse signal (SPLADE) is extremely strong. However, for broader concepts like “Attention” (which has polysemous meanings in Psychology vs. CS), the dense vector plays a larger role in guiding the ranking, resulting in a lower overlap with the pure keyword search (23%).

4.3 Case Studies

To demonstrate the system’s capability to track semantic evolution, we conducted analyses on some terms that have significant historical shifts: “Transformer”, “Entropy”, and “Token”.

4.3.1 Transformer. The term “Transformer” represents a classic case of semantic drift where a new meaning completely eclipses the old one. Historically, it was commonly associated with a passive component in Electrical Engineering that transfers electrical energy. One may be bold enough to claim that it has an equivalent or nearly-relevant association with integral transforms in mathematics (e.g., Laplace transform).

In late 2017, following the publication of “Attention Is All You Need” [28], the term was co-opted by the NLP community to describe a deep learning architecture. Our analysis reveals three distinct trends. First, the mathematical concept of transformers maintains a steady, baseline relevancy throughout the time period. Second, we observe a gained relevancy in electrical engineering concepts starting around 2006. Finally, and most prominently, the deep learning interpretation explodes in relevancy in the 2019–2020 period. This surge corresponds to the widespread adoption of pre-trained models like BERT [6] and the onset of the GPT [24] era, where the “Transformer” became the term that backbones the current state of natural language processing.

The system identified the following key eras for the Deep Learning cluster (Figure 3):

- **2016–2019**: 0.2 papers/year
- **2020**: 9.0 papers/year
- **2021–2022**: 19.5 papers/year
- **2023**: 45.0 papers/year
- **2024–2025**: 64.5 papers/year

4.3.2 Entropy. For “Entropy”, the system identifies a two concepts between the physical and information sciences. In the context of physics and thermodynamics, the concept maintains a steady, consistent relevancy over time, as expected for a foundational physical law. However, a distinct divergence is observed starting in 2008, where the concept’s application in Information Theory begins to grow significantly. This growth is likely correlated with the renaissance of deep learning [12], where concepts like cross-entropy became directly associated with loss functions for training neural networks. The system effectively captures how a term can expand its semantic footprint without losing its original meaning.

Key eras identified for the Information Theory cluster include (Figure 4):

- **1993–2005**: 1.6 papers/year
- **2006–2011**: 10.2 papers/year
- **2012–2020**: 20.2 papers/year

Table 4: Jaccard Similarity of Top-50 Results

Query	Dense ∩ Sparse	Dense ∩ Hybrid	Sparse ∩ Hybrid
Transformer	0.00	0.01	0.75
Corona	0.00	0.00	0.75
Entropy	0.02	0.15	0.12
Token	0.03	0.09	0.16
Attention	0.06	0.31	0.23
Neural Networks	0.03	0.16	0.11
Diffusion	0.01	0.04	0.41

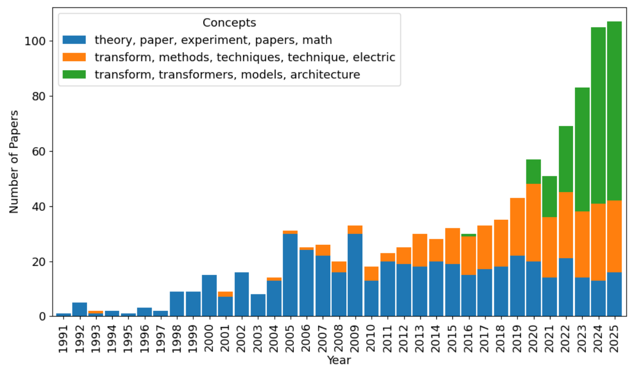

Figure 3: "Transformer" concept frequency over time (α = 0.75)

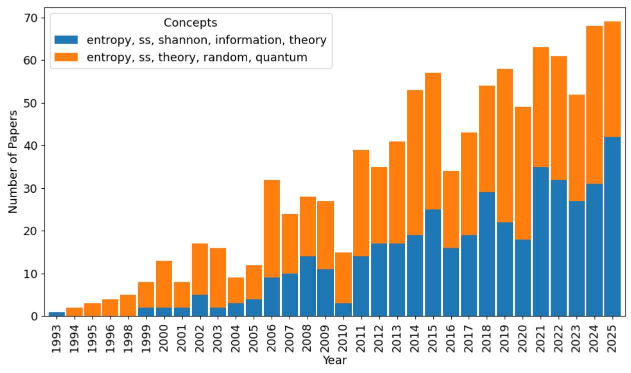

Figure 4: "Entropy" concept frequency over time

- **2021–2024**: 31.2 papers/year
- **2025**: 42.0 papers/year

4.3.3 Token. The term "Token" has recently split into two very different meanings, showing how a single word can simultaneously be adopted by two rapidly growing (yet distinct) technological fields. Post-2016, we see a massive surge in relevancy related to language models and natural language processing. This aligns with the industry's shift from word-level embeddings like Word2Vec [18] to sub-word tokenization strategies used in modern architectures [6]. Then, in post-2017, a secondary but distinct cluster emerges related to cryptography and blockchain technologies (e.g., NFTs). SCET proves its value here by easily distinguishing between these two meanings, even though they both exploded in popularity at the same time.

SCET was able to identify the explosion of the NLP cluster in the following time periods (Figure 5):

- **1994–2019**: 1.2 papers/year
- **2020–2021**: 22.5 papers/year
- **2022–2023**: 70.0 papers/year
- **2024**: 239.0 papers/year
- **2025**: 412.0 papers/year

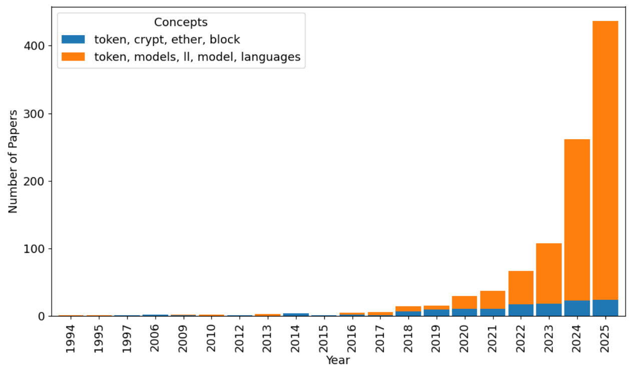

Figure 5: "Token" concept frequency over time

4.4 Qualitative Analysis of Pivotal Papers

A key feature of SCET is the identification of "Pivotal Papers"—documents that are semantically central to a concept during a specific era. Qualitatively, we observed that the system tends to select two types of papers as pivotal:

(1) **Foundational Methodologies**: For the "Transformer" (NLP) era starting in 2017, the system was not able to correctly identify *Attention Is All You Need* [28] as a pivotal paper, but it did identify *Set Transformer: A Framework for Attention-based Permutation-Invariant Neural Networks* [13], which is a related paper that cites [28]. This confirms that the hybrid score is able to capture the seminal nature of the work.

(2) **Comprehensive Reviews**: In mature eras (e.g., "Deep Learning" in 2015-2018), the system often surfaces highly-cited review papers (e.g., *Deep Learning* by LeCun, Bengio, and Hinton [12]). These papers sit at the "center" of the cluster because they aggregate terminology from across the entire subfield, making them semantically representative of the era.

5 Discussion

5.1 The Time Machine Effect and Burstiness

The core contribution of SCET is its ability to function as a "time machine" for scientific semantics, capturing the non-linear nature of scientific progress. Traditional bibliometric analysis often relies on fixed temporal buckets (e.g., decade-by-decade) or sliding windows. However, one may claim that science can be characterized by "burstiness", that is, long periods of incremental progress with rapid paradigm shifts. Fixed windows tend to smooth out these bursts, obscuring the exact moment of change.

Our use of Decision Tree Regression addresses this by allowing the data to define the boundaries. Because a decision tree splits data based on minimizing variance, it naturally places boundaries at the points of maximum change. This allows SCET to capture the "sharp edges" of scientific history, such as the sudden emergence of COVID-19 literature in early 2020 or the immediate impact of *Attention Is All You Need* in late 2017. By quantifying these shifts, we can potentially predict the emergence of new subfields. For instance, a sudden increase in the variance of a concept's embedding cluster might indicate a split of consensus or the birth of a new application domain.

5.2 Scalability vs. Depth Trade-off

Our benchmarks highlight a fundamental trade-off between interactivity and analytical depth. The clustering step is $O(N \cdot K \cdot D)$, where N is the number of papers, K is clusters, and D is vector dimension (768). With $N = 1000$, the system is responsive (2–3s). However, limiting the analysis to the top 1000 papers means we might miss subtle, long-running evolutionary trends that appear in lower-ranked documents. Increasing N to 5000 improves the semantic resolution but pushes latency to near 20s, breaking the interactive feedback loop.

A potential solution for future work is to implement a *hierarchical* approach: perform a fast, coarse-grained clustering on a large dataset using a dimensionality-reduced vector space (for example, using PCA to reduce 768d to 64d), and then perform fine-grained clustering on the user's region of interest. Alternatively, pre-computing clusters for common queries could offload the cost, though this sacrifices the flexibility of dynamic, query-specific concept discovery. Another avenue is to leverage GPU acceleration for the K-Means step (using libraries like RAPIDS cuML), which could likely reduce the clustering time by an order of magnitude, allowing for $N = 10000+$ in real-time.

5.3 Interpretability in Vector Retrieval

A significant challenge in deploying vector-based systems is the "Black Box" nature of dense embeddings. When a user queries for "Transformer" and receives a paper about "Self-Attention" that does not contain the word "Transformer", the retrieval is semantically correct but potentially confusing. SCET addresses this interpretability gap through its hybrid architecture. By maintaining the sparse SPLADE vectors alongside the dense SPECTER2 vectors, we can generate human-readable explanations for the clusters. The top-weighted tokens in the sparse vector serve as a generated label for the semantic cluster. This is particularly valuable for interdisciplinary researchers who may not be familiar with the specific jargon of a new field.

5.4 Limitations

While effective, the current system has several limitations that warrant discussion.

5.4.1 *Citation Lag and Impact Metrics.*

Identifying pivotal papers in the current year is inherently difficult because they have not yet accumulated citations, which is the standard metric for impact. SCET relies on semantic centrality (hybrid score), which is a leading indicator, but it may bias results towards papers that use standard terminology rather than those that introduce radical new terms. A paper that coins a completely new term might initially appear as an outlier in the vector space, rather than a centroid. Integrating a "novelty score" that rewards semantic distinctiveness could address this.

5.4.2 *Vocabulary Mismatch and Rank Fusion.*

The ablation study observed that for some queries, the dense and sparse retrievals are orthogonal. While our weighted sum ($\alpha = 0.5$) attempts to bridge this, it is a naive linear combination. A more sophisticated rank fusion algorithm, such as Reciprocal Rank Fusion (RRF), might better handle cases where the two lists are disjoint. RRF ranks documents based on the inverse of their rank in each individual list, ensuring that a document that is highly ranked in *either* list is preserved, rather than being diluted by a low score in the other.

5.4.3 *Data Source and Language Bias.*

The system is currently limited to arXiv metadata. This introduces a significant domain bias towards Physics, Mathematics, and Computer Science. Concepts in Biology, Medicine, or the Humanities are underrepresented. Furthermore, the models (SPECTER2 and SPLADE) are primarily trained on English scientific text. This limits the system's ability to track global scientific evolution, particularly in regions where non-English publication is common. Expanding the ingestion pipeline to include PubMed Central or OpenAlex would provide a more holistic view of science.

5.4.4 *Granularity of Analysis.*

Currently, SCET operates on abstracts. While abstracts are dense summaries, they often omit the specific methodological details or experimental results that define a paper's contribution. A term might appear in the abstract but be used in a different context in the body; full-text ingestion would allow for more granular analysis. However, this would increase the storage and compute requirements by several orders of magnitude.

5.5 Future Work

Beyond addressing the limitations, there are several exciting directions for future research.

5.5.1 Redundant Cluster Labels. A frequent observation in the clustering results is the redundancy of cluster labels relative to the search query. For instance, when querying for "Transformer", one of the resulting clusters is often labeled with terms like "transform", "transformation", or "transformers". This occurs because the labeling mechanism extracts the most representative keywords from the documents in each cluster. Since the documents were retrieved based on the query, the query terms are naturally the most statistically significant features in the text. While semantically accurate, these labels provide low information gain to the user. Future iterations of the pipeline should implement a dynamic stop-word list that filters out the query terms and their variants during the cluster labeling phase to only show more descriptive, sub-topic specific keywords.

5.5.2 Multi-modal Analysis. Scientific papers are not just text; they contain figures, equations, and tables. Integrating multi-modal embeddings (like CLIP [23] for figures) could allow users to track the evolution of visual concepts (e.g., neural network architecture diagrams).

5.5.3 Citation Graph Integration. While SPECTER2 implicitly uses citations, explicitly using the citation graph during the clustering phase could improve the coherence of the identified sub-concepts.

5.5.4 User Feedback Loop. Implementing a relevance feedback mechanism where users can correct the clustering or era boundaries would allow the system to learn and adapt to specific domain nuances over time. This could be implemented as an Active Learning loop, where the user's corrections are used to fine-tune a lightweight adapter layer on top of the frozen SPECTER2 embeddings, effectively personalizing the semantic space for specific research groups.

5.5.5 Cross-Lingual Analysis. Extending the system to support non-English papers would require replacing SPECTER2 with a multilingual model such as mBERT [5]. This would enable the tracking of concepts as they migrate across language barriers, potentially revealing interesting lags in global knowledge transfer.

6 Conclusion

In this work, we presented the Scientific Concept Evolution Tracker (SCET), a scalable system for quantifying and visualizing the semantic drift of scientific terms. By integrating a hybrid vector search engine (Milvus) with a relational metadata store (PostgreSQL), we achieved a system that is both powerful in semantics and lexically precise. Our methodology of combining dynamic K-Means clustering with temporal segmentation allows users to inherently discover distinct sub-concepts and their active eras with convenience.

Our experimental evaluation demonstrates that SCET is capable of interactive performance, processing complex queries over thousands of documents in under 4 seconds. The ablation study highlighted the critical importance of the hybrid retrieval strategy, revealing that semantic and lexical searches often return disjoint sets of results for scientific terminology. Case studies on "Transformer", "Entropy", "Token" confirmed the system's ability to correctly disambiguate similar terms and identify historical turning points. SCET is a notable exploration into the field of the "Science of Science", and serves as a tool for users to navigate the ever-expanding corpus of scientific literature via both semantic and temporal methods.

Acknowledgments

I thank Dr. Nick Koudas for his guidance and support throughout CSC2508 Advanced Data Systems.

References

[1] arXiv.org submitters. 2024. arXiv Dataset. doi:10.34740/KAGGLE/DSV/7548853

[2] Robert Bamler and Stephan Mandt. 2017. Dynamic Word Embeddings. arXiv:1702.08359 [stat.ML] https://arxiv.org/abs/1702.08359

[3] Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: Pretrained Language Model for Scientific Text. In *EMNLP*. arXiv:arXiv:1903.10676

[4] Arman Cohan, Sergey Feldman, Iz Beltagy, Doug Downey, and Daniel S. Weld. 2020. SPECTER: Document-level Representation Learning using Citation-informed Transformers. arXiv:2004.07180 [cs.CL] https://arxiv.org/abs/2004.07180

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR* abs/1810.04805 (2018). arXiv:1810.04805 http://arxiv.org/abs/1810.04805

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805 [cs.CL] https://arxiv.org/abs/1810.04805

[7] Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. 2021. SPLADE: Sparse Lexical and Expansion Model for First Stage Ranking. arXiv:2107.05720 [cs.IR] https://arxiv.org/abs/2107.05720

[8] Cloud Native Computing Foundation. 2022. etcd v3.5.5. https://github.com/etcd-io/etcd

[9] G. W. Furnas, T. K. Landauer, L. M. Gomez, and S. T. Dumais. 1987. The vocabulary problem in human-system communication. *Commun. ACM* 30, 11 (Nov. 1987), 964–971. doi:10.1145/32206.32212

[10] PostgreSQL Global Development Group. 2025. What is PostgreSQL? https://www.postgresql.org/docs/15/intro-whatis.html

[11] Carlos Lassance, Hervé Déjean, Thibault Formal, and Stéphane Clinchant. 2024. SPLADE-v3: New baselines for SPLADE. arXiv:2403.06789 [cs.IR]

[12] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521, 7553 (2015), 436–444. doi:10.1038/nature14539

[13] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam R. Kosiorek, Seungjin Choi, and Yee Whye Teh. 2019. Set Transformer: A Framework for Attention-based Permutation-Invariant Neural Networks. arXiv:1810.00825 [cs.LG] https://arxiv.org/abs/1810.00825

[14] Yu. A. Malkov and D. A. Yashunin. 2018. Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs. arXiv:1603.09320 [cs.DS] https://arxiv.org/abs/1603.09320

[15] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press, USA.

[16] Lucas Maystre, Alvaro Ortega Gonzalez, Charles Park, Rares Dolga, Tudor Berariu, Yu Zhao, and Kamil Ciosek. 2025. When Embedding Models Meet: Procrustes Bounds and Applications. arXiv:2510.13406 [cs.LG] https://arxiv.org/abs/2510.13406

[17] Dirk Merkel. 2014. Docker: lightweight Linux containers for consistent development and deployment. *Linux J.* 2014, 239, Article 2 (March 2014).

[18] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. arXiv:1301.3781 [cs.CL] https://arxiv.org/abs/1301.3781

[19] Milvus. 2025. IVF_PQ | Milvus Documentation. https://milvus.io/docs/ivf-pq.md [Online; accessed 13-December-2025].

[20] MinIO. 2023. MinIO. https://github.com/minio/minio

[21] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Alessandro Moschitti, Bo Pang, and Walter Daelemans (Eds.). Association for Computational Linguistics, Doha, Qatar, 1532–1543. doi:10.3115/v1/D14-1162

[22] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. arXiv:1802.05365 [cs.CL] https://arxiv.org/abs/1802.05365

[23] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning Transferable Visual Models From Natural Language Supervision. arXiv:2103.00020 [cs.CV] https://arxiv.org/abs/2103.00020

[24] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. (2018).

[25] Stephen Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. 1994. Okapi at TREC-3. 0–.

[26] Gerard Salton, Edward A. Fox, and Harry Wu. 1983. Extended Boolean information retrieval. *Commun. ACM* 26, 11 (Nov. 1983), 1022–1036. doi:10.1145/182.358466

[27] Amanpreet Singh, Mike D'Arcy, Arman Cohan, Doug Downey, and Sergey Feldman. 2022. SciRepEval: A Multi-Format Benchmark for Scientific Document Representations. In *Conference on Empirical Methods in Natural Language Processing*. https://api.semanticscholar.org/CorpusID:254018137

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

[29] Jianguo Wang, Xiaomeng Yi, Rentong Guo, Hai Jin, Peng Xu, Shengjun Li, Xiangyu Wang, Xiangzhou Guo, Chengming Li, Xiaohai Xu, Kun Yu, Yuxing Yuan, Yinghao Zou, Jiquan Long, Yudong Cai, Zhenxiang Li, Zhifeng Zhang, Yihua Mo, Jun Gu, Ruiyi Jiang, Yi Wei, and Charles Xie. 2021. Milvus: A Purpose-Built Vector Data Management System. In *Proceedings of the 2021 International Conference on Management of Data* (Virtual Event, China) *(SIGMOD '21)*. Association for Computing Machinery, New York, NY, USA, 2614–2627. doi:10.1145/3448016.3457550

[30] Wikipedia contributors. 2025. Small-world experiment — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Small-world_experiment&oldid=1320552337 [Online; accessed 12-December-2025].