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 "Science of Science" is understanding how scientific consensus and terminologies evolve. 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.

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 Information Retrieval 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 [21].

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 [12].

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 [12].

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 [20].

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 [15] and GloVe [18] 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 [19] and BERT [5] introduced dynamic embeddings to address the limitations of static models. The Transformer [24] 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 [12].

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 [26]. 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.

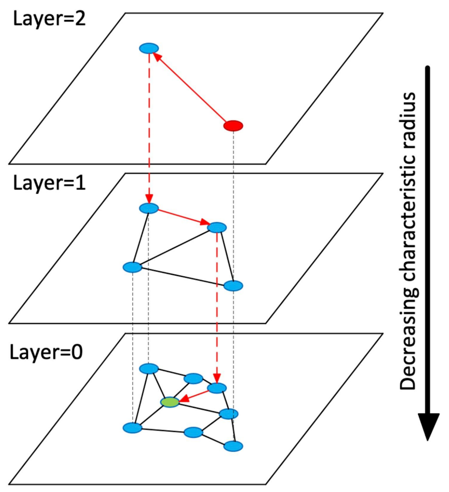

Figure 1: Visualization of HNSW [23]

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 1. 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 [11].

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 [13] .

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.

This structured metadata is stored in a PostgreSQL [9] 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 [25] 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 latency at the cost of higher memory consumption, compared to quantization-based indexes like Inverted File with Product Quantization [16]. 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 [14] 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 [7] for metadata management and service discovery, MinIO [17] as an S3-compatible object storage service for persisting vector data, and PostgreSQL for storing structured paper metadata.

The system is deployed on a Linux-based home lab server. The hardware consists of an Intel Xeon E5-1620 v3 CPU (8 cores @ 3.50 GHz), 32GB of RAM, and a Seagate BarraCuda 20TB HDD.

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

Resource limits are enforced via Docker Compose, with Milvus allocated 16GB of RAM, etcd 2GB, and MinIO 8GB.

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”) [8]. 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 [22], 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) [10]. 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 [6].

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 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 [1] over a range such as $[2, 8]$.

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. A naive approach might look for peaks or valleys, but publication

[1]The Silhouette Score measures how similar an object is to its own cluster (cohesion) compared to other clusters (separation), which is a metric we can use for cluster quality without the need for the actual labels.

data is noisy. Instead, we employ a Decision Tree Regressor to approximate this signal with a step function. By fitting a decision tree to the annual publication count with a limited depth, 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, 2000–2012: low activity, 2013–2023: high activity). 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, highlighting the seminal works that defined each era.

4 Experimental Results

4.1 System Performance Benchmarks

4.1.1 End-to-End Pipeline Latency.

4.1.2 Clustering Scalability.

4.2 Case Studies

4.2.1 Transformer.

4.2.2 Corona.

4.3 Ablation Study: Hybrid vs. Dense-Only

5 Discussion

5.1 The Time Machine Effect

5.2 Scalability vs. Depth Trade-off

5.3 Limitations

6 Conclusion

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. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805 [cs.CL] https://arxiv.org/abs/1810.04805

[6] 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

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

[8] 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

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

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

[11] 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

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

[13] 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

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

[15] 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

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

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

[18] 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

[19] 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

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

[21] 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

[22] 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

[23] Stefan Webb. 2025. Understanding Hierarchical Navigable Small Worlds (HNSW) for Vector Search. https://milvus.io/blog/understand-hierarchical-navigable-small-worlds-hnsw-for-vector-search.md [Online; accessed 14-December-2025].

[24] 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

[25] 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

[26] 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].