Hierarchical Semantic Retrieval with Cobweb

Anant Gupta* AGUPTA886@GATECH.EDU
Karthik Singaravadivelan* KSINGARA3@GATECH.EDU
Zekun Wang ZEKUN@GATECH.EDU
College of Computing, Georgia Institute of Technology, Atlanta, GA 30332 USA

Abstract

Neural document retrieval often treats a corpus as a flat cloud of vectors scored at a single granularity, leaving corpus structure underused and explanations opaque. We use Cobweb–a hierarchy-aware framework–to organize sentence embeddings into a prototype tree and rank documents via coarse-to-fine traversal. Internal nodes act as concept prototypes, providing multi-granular relevance signals and a transparent rationale through retrieval paths. We instantiate two inference approaches: a generalized best-first search and a lightweight path-sum ranker. We evaluate our approaches on MS MARCO and QQP with encoder (e.g., BERT/T5) and decoder (GPT-2) representations. Our results show that our retrieval approaches match the dot product search on strong encoder embeddings while remaining robust when kNN degrades: with GPT-2 vectors, dot product performance collapses whereas our approaches still retrieve relevant results. Overall, our experiments suggest that Cobweb provides competitive effectiveness, improved robustness to embedding quality, scalability, and interpretable retrieval via hierarchical prototypes.

1. Introduction

Humans naturally organize knowledge into hierarchies and reason with prototypes: representative exemplars that capture the central tendency of a concept while allowing graded membership and basic-level advantages in categorization (Rosch & Mervis, 1975; Rosch, 1988). These cognitive principles suggest that effective information access should benefit from *both* hierarchical structure (topics → subtopics) and prototype-based reasoning (representative exemplars that summarize clusters). Classic conceptual clustering methods, most notably COBWEB (Fisher, 1987), formalized these ideas by incrementally constructing a classification tree whose internal nodes summarize data with concept-level statistics and whose leaves capture finer distinctions.

In large-scale document retrieval, modern systems have shifted from lexical matching to neural embedding methods. BM25 is a sparse lexical ranking function that scores query–document matches using term frequency, inverse document frequency, and document-length normalization (Robertson & Zaragoza, 2009). More recent neural approaches learn contextual representations that capture semantics with the Transformer architecture (Vaswani et al., 2023): cross-encoders (e.g., BERT re-rankers) model full query–document interactions for high accuracy (Nogueira & Cho, 2019), and dual-encoder dense retrievers map both into a shared vector space for scalable nearest-

∗. Equal contribution.

This work is licensed under a Creative Commons Attribution International 4.0 License.

neighbor search (Karpukhin et al., 2020; Reimers & Gurevych, 2019). Late-interaction models such as ColBERT further improve effectiveness vs. efficiency trade-offs by matching token-level representations (Khattab & Zaharia, 2020).

Despite their popularity, most neural retrievers treat the corpus as a flat cloud of points in Euclidean space, with relevance computed by a single similarity at a single granularity. However, flat retrievers underutilized the corpus' inherent topic structure; hierarchical indexes such as HNSW (Malkov & Yashunin, 2020) enable coarse-to-fine retrieval by searching from upper to lower layers. Prototype-based reasoning can provide concept-level anchors that make ranked results and re-ranking decisions interpretable without sacrificing performance (Anand et al., 2022).

Our work bridges the gaps between recent neural document retrieval approaches and human-like interpretability with Cobweb, a prototype- and hierarchy-aware retrieval framework that learns a hierarchical "database" over document embeddings. Cobweb incrementally organizes the corpus into a tree whose internal nodes store intermediate prototypes that summarize their descendants. At query time, retrieval proceeds coarse-to-fine: the query is matched to high-level prototypes and then refined down the tree to leaf documents. Central to this process is a hierarchical retrieval mechanism that aggregates prototype similarities along the query's traversal path, yielding multi-granular relevance signals and an auditable explanation.

Our contributions are the following. 1. We introduce hierarchical retrieval approaches that compose coarse-to-fine prototype similarities along a learned tree using Cobweb, generalizing flat vector-space matching to multi-level relevance and enabling efficient search using learned sentence embeddings from neural models. 2. We show how learning intermediate prototypes improves interpretability by exposing concept-level rationales. 3. We ground our approach on the MS MARCO (Bajaj et al., 2018) and QQP (Wang et al., 2018) datasets, using encoder-only, decoder-only, and encoder–decoder Transformer embeddings. Our hierarchical retrieval methods match or improve upon inner-product-based dense-retrieval baselines across embedding architectures, scale efficiently with both time complexity and data size, and yield human-interpretable prototype paths.

2. Related Works

2.1 Language Representations

Language representations aim to encode linguistic meaning in a form usable by learning algorithms. Early distributional semantics approaches were centered around sparse and symbolic bag-of-words with TF–IDF weighting and n-gram language models. These approaches captures term frequency and short-range co-occurrence but ignoring broader context and treating words as independent types.

Recent embedding-based approaches address these limitations by mapping tokens to dense vectors whose geometry reflects meaning. Work on Word2Vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014) showed that words can be embedded in a continuous space where proximity correlates with semantic similarity. The introduction of Transformers (Vaswani et al., 2023) enabled contextualized embeddings via self-attention, substantially improving performance across natural language processing tasks such as sentiment analysis and machine translation.

Encoder-only Transformer models such as BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019) produce bidirectional representations that are widely used for retrieval. Decoder-only Trans-

former models such as GPT (Radford et al., 2019) are primarily designed for generation but can provide usable embeddings from internal activations. Encoder–decoder architectures, such as T5 (Raffel et al., 2023), handle sequence-to-sequence tasks while producing high-quality contextual encodings. In this work, we compare and contrast our document retrieval approaches using the embeddings produced by these three types of Transformer models.

2.2 Document Retrieval

Document retrieval ranks a large corpus by relevance to a user query, typically using a retriever to produce top-k candidates. Classical sparse lexical methods such as BM25 (Robertson & Zaragoza, 2009) rank documents by counting query-term matches, weighting rare terms more, and adjusting for repeated terms and document length, favoring exact lexical overlap. Dense neural retrieval encodes queries and documents as continuous vectors and retrieves by nearest-neighbor search under inner product or cosine similarity (Karpukhin et al., 2020); late-interaction models such as ColBERT (Khattab & Zaharia, 2020) retain token-level granularity while preserving efficient dot-product search.

At scale, nearest-neighbor search relies on specialized indexes which are optimized for this dot product objective deeply ingrained into most embeddings models. FAISS (Johnson et al., 2017) provides exact (e.g., `IndexFlatIP`) and approximate indexes for dot-product search, and has been an industry standard owing to its robust implementation and matrix-operable parallelization. Graph-based, approximate nearest-neighbor structures like HNSW (Malkov & Yashunin, 2020) enable coarse-to-fine exploration. These measures are effective because they utilize an inner-product calculation (usually the dot-product). However, these systems also largely treat the corpus as a flat set of vectors and rely on a single-step geometric similarity. In contrast, our approach organizes documents into a semantic hierarchy with interpretable prototypes, yielding multi-granular relevance signals and transparent retrieval paths that complement flat nearest-neighbor search and reranking.

2.3 Hierarchical Clustering with Cobweb

Hierarchical clustering organizes data into nested groups, such as a tree or a directed acyclic graph, that support analysis at multiple levels of granularity. Early agglomerative methods (Sneath, 1957; Ward, 1963) construct such structures by iteratively merging items under a distance metric and linkage criterion, without learned prototypes or online adaptation.

Cobweb (Fisher, 1987) instead performs incremental hierarchical clustering on concepts: it maintains a probabilistic taxonomy whose nodes summarize attribute distributions, and inserts each instance by maximizing category utility with create/merge/split/reorder operators. Because of Cobweb's nature of sorting top-down, it can be thought of as an unsupervised divisive strategy for concept formation, in contrast with agglomerative methods. Extensions to Cobweb (MacLellan et al., 2022; MacLellan & Thakur, 2021; Wang et al., 2025) brought Cobweb forward as a substitute for neural applications and used its underlying architecture to imagine neural approaches, and recent studies show Cobweb's robustness in vision and language tasks (Barari et al., 2024b,a; Lian et al., 2025). In this work, we repurpose Cobweb as a hierarchical database for document retrieval:

internal nodes serve as interpretable prototypes that guide coarse-to-fine search and provide multi-granular relevance signals, offering a complementary alternative to flat nearest-neighbor retrieval.

3. Methodology

Our goal is to adapt Cobweb for large-scale semantic document retrieval, where documents and queries are represented not by discrete lexical features but by continuous, high-dimensional vectors. Modern neural retrieval methods rely on such *latent semantic representations*, mapping text into a continuous space in which geometric proximity reflects semantic relatedness. This transformation is essential for moving beyond the lexical surface form: two documents discussing "car insurance" and "auto coverage" may share few or no exact terms, yet must be recognized as near-equivalent in meaning.

We use the Cobweb/4V (Barari et al., 2024b) algorithm for our experiments, which extends the original Cobweb incremental concept formation framework to support continuous features. However, applying Cobweb/4V in this setting introduces two challenges. First, prototypes within Cobweb/4V use diagonal Gaussian distributions, assuming that input features are conditionally independent. This property is not guaranteed for neural embeddings optimized on the dot product, which often contain correlated dimensions. Second, the Cobweb/4V algorithm is designed to return a best-match concept that aggregates all retrieved nodes (including both intermediate and leaf ndoes) rather than returning all of the retrieved leaf nodes. Our methodology addresses both limitations: we preprocess embeddings with whitening to improve feature independence that can be captured by a diagonal variance structure, and we extend a new inference for Cobweb/4V to return multi-result ranking over its learned hierarchy.

3.1 Sentence Representations

For Cobweb/4V to organize and search effectively, document vectors must capture high-level semantic relationships and yield interpretable prototypes that summarize coherent concepts. To achieve this, we use sentence embeddings derived from pre-trained Transformer-based language models that encode contextual meaning into dense vectors.

Specifically, we treat a tokenized sentence $x_{1:T} = (x_1, \ldots, x_T)$ as input and obtain a vector from a pre-trained Transformer f_θ by pooling the representation of the final hidden states: $z = g(f_\theta(x_{1:T})) \in \mathbb{R}^d$, where f_θ has fixed (pre-trained) parameters θ, g is a model-specific pooling function, and z is the sentence embedding of dimension d.

We evaluate three representative Transformer families: BERT (Devlin et al., 2019) (encoder-only), GPT-2 (Radford et al., 2019) (decoder-only), and T5 (Raffel et al., 2023) (encoder–decoder). While GPT-2 is primarily optimized for autoregressive generation rather than isotropic semantic embeddings, we include it to examine how generative-model representations perform in our retrieval framework and to serve as a contrasting baseline to embedding-oriented architectures. For BERT and T5, we test sentence-transformer variants (Reimers & Gurevych, 2019), which fine-tune encoders with a contrastive objective that better aligns embeddings with semantic similarity and perform better on similarity objectives. Further details of the model architectures and training objectives are provided in Appendix A.

3.2 Dimensional Independence of Representations

Neural sentence embeddings, while effective for capturing semantics, often exhibit strong correlations across dimensions due to the way they are learned (Li et al., 2020). However, Cobweb/4V assumes a diagonal covariance structure for the Gaussian parameters at each node. To address this, we apply *embedding whitening* techniques, including Principal Component Analysis (PCA) and Independent Component Analysis (ICA) (Yamagiwa et al., 2023). PCA not only helps decorrelate features but also allows optional dimensionality reduction, which can improve computational efficiency in large-scale retrieval settings. ICA further enhances isotropy—making the distribution more uniform and spherical—while reducing residual cross-dimensional correlations. By producing representations that can be better captured by a diagonal covariance structure, whitening ensures that the algorithm's probabilistic category utility computations remain meaningful and stable in a dense-vector setting.

Although numerous other whitening methods exist, we focus on these relatively simple techniques to isolate and evaluate the general efficacy of whitening in our retrieval framework.

3.3 Cobweb/4V Training

Cobweb/4V incrementally constructs a hierarchy in which each internal node stores a *prototype* as parameterized by Gaussian parameters μ and σ^2 that summarizes the instances (documents) beneath it. These prototypes are updated online as a new document inputs, and the Cobweb/4V selects among four possible operations at each step—create a new category, merge categories, split a category, or insert into an existing category using the information-theoretic *category utility* (CU) metric (Corter & Gluck, 1992):

$$CU(c) = P(c)\left[U(c_p) - U(c)\right],$$

where $U(c)$ denotes the entropy of concept c computed over its attribute distributions, and c_p is the parent of node c. This measure balances category predictiveness and distinctiveness, guiding the construction of meaningful, discriminative prototypes.

After training, all raw document vectors extracted from language models are stored only at the leaf nodes, and each leaf corresponds to exactly one datapoint from the training set. Internal nodes do not store raw instances; instead, they maintain the probabilistic prototypes summarizing their descendants (the Gaussain parameters μ_c and σ^2 over sentence embeddings). This design ensures that retrieval can always access the original datapoints while still benefiting from the generalization properties of the internal prototypes.

The resulting hierarchy can be viewed as a hierarchical clustered database that encodes both fine-grained document instances and high-level semantic abstractions at internal nodes. We hypothesize that, when combined with appropriately whitened embeddings, this structure can improve prediction and retrieval accuracy by exploiting both local and global semantic organization.

3.4 Hierarchical Prediction

Once trained, the learned Cobweb/4V hierarchy functions as both a semantic index and a search structure over the embedding space. We devise and evaluate two prediction strategies.

Given a query vector $x \in \mathbb{R}^d$, both retrieval methods rely on the *collocation score* (Jones, 1983): $s(c) = p(x \mid c)\, p(c \mid x)$, where c is a candidate concept node with Gaussian parameters μ_c and σ_c^2. This score reflects both how well the query matches the node's prototype $p(x \mid c)$ and how representative the node is of the query's inferred category $p(c \mid x)$. In this paper, we assume a uniform prior $p(c)$ over all concepts for a faster computation of $s(c)$. Specifically, applying Bayes' rule: $p(c \mid x) = \frac{p(x|c)\, p(c)}{\sum_{c'} p(c')\, p(x|c')}$, the collocation score simplifies to:

$$s(c) = N \cdot p(x \mid c)^2,$$

where N is a normalization constant independent of c. Thus, ranking by $s(c)$ is equivalent to ranking by $p(x \mid c)$ alone. We exploit this simplification in both prediction methods described below.

3.4.1 Generalized Best-First Search

We generalize the original Cobweb prediction methodology, which only predicts the single best element. Retrieval begins at the root and performs a greedy best-first search through the hierarchy using $s(c)$ as the heuristic. Instead of committing to a single greedy descent path, Cobweb/4V expands up to N_{max} nodes according to their collocation scores, enabling exploration of multiple promising branches. Let $\mathcal{C}^*$ be the collection of nodes as the result of expansion, and since documents are stored in the leaves, we rank the leaves in the order they are explored in $\mathcal{C}^*$ and return them.

3.4.2 Path Sum Prediction

In this alternative approach, leaves are ranked by their cumulative path collocation scores. For a candidate leaf ℓ, we define

$$\text{score}(\ell) = \sum_{i=1}^{|\text{path}(\ell)|} \log\big(s(c_i)\big),$$

where $c_1, \ldots, c_{|\text{path}(\ell)|}$ are the internal nodes on the path from the root to ℓ. Unlike the generalized best-first search approach, which orders documents by the sequence in which nodes are expanded, path-sum prediction ranks documents directly by their path scores.

4. Experiment

4.1 Experimental Setup

Datasets. We evaluated our approach and baselines on two retrieval tasks: MS MARCO (Bajaj et al., 2018) and QQP (Wang et al., 2018). MS MARCO is a large-scale web search corpus built from real Bing queries and web content. We use its passage collection as our document pool, which contains 8.8M passages and 7k testing queries. We use it in an ad hoc retrieval setup: given a natural-language query (e.g., "how to clean a laptop keyboard"), rank passages from the document pool by relevance and return the top-k results. QQP is a paraphrase identification dataset of paired Quora questions. It contains 364k training pairs, 40k validation pairs, and 391k test pairs. We recast

it as a retrieval task where, for a given query question (e.g., "How can I become a better cook?"), the objective is to retrieve its paraphrase as the relevant "document" from a candidate pool.

Metrics. We report Recall@x, Mean Reciprocal Rank (MRR@x), and Normalized Discounted Cumulative Gain (nDCG@x) for $x \in \{5, 10\}$ on both MS MARCO and QQP, following prior work (Khattab & Zaharia, 2020). Mean Reciprocal Rank (MRR) computes the reciprocal of the rank of the first relevant document for each query, averaged across queries. Higher values indicate that at least one relevant document appears closer to the top of the ranking. Additionally, Normalized Discounted Cumulative Gain (nDCG) accounts for both document relevance and position, rewarding highly relevant documents that appear earlier. Scores are normalized so that 1 indicates a perfect ranking, making nDCG particularly useful when relevance is graded.

4.2 Compared Approaches and Implementation

Baselines. To effectively test the dot product, we use Facebook AI Similarity Search (FAISS) as our baseline, a library which optimizes exact-nearest-neighbors search by dot product through matrix-operable parallelization. Specifically, we employ the flat index (`IndexFlatIP`), which performs an exact dot-product search. This serves as a high-accuracy baseline for efficient similarity search over vector embeddings.

Cobweb/4V Variants. For each transformer model, we train two Cobweb/4V models as described in Section 3.3. One model uses raw transformer embeddings, the other uses PCA and ICA whitened embeddings. We evaluate two prediction methods on these hierarchies: **Cobweb-BFS** that performs the best-first Search method described in Section 3.4.1 and **Cobweb-PathSum** that performs path sum prediction as described in Section 3.4.2.

Shared Implementation Details. In all experiments, queries and candidate documents are embedded using the language models described in Section 3.1, and retrieval is performed in the resulting embedding space. For Cobweb variants, embeddings are whitened prior to tree construction, with an explained variance threshold of 0.96 used throughout. For dataset construction, we select a corpus of n documents and q queries such that each query has at least one corresponding answer document in the corpus.

Architecture Comparison. To compare retrieval effectiveness across model architectures, we evaluate all approaches on a controlled setting with a fixed corpus of 10k documents and 1k queries for both QQP and MS MARCO. This setup allows us to isolate the impact of different embedding models (RoBERTa, GPT-2, T5) and retrieval strategies (Dot Product, Cobweb-BFS, Cobweb-PathSum) without confounding effects from corpus size. The goal of this experiment is to assess how our retrieval approaches perform relative to a strong flat index baseline across a variety of embedding geometries.

Scaling Experiments. To study robustness at larger scales, we fix the embedding model architecture and vary the corpus and query sizes. By separating the architecture comparison from scaling analysis, we first evaluate our retrieval approaches' ability to leverage sentence embeddings in a fair,

fixed-size regime, and then assess how well those gains hold when moving to larger, more realistic datasets.

5. Results and Discussion

5.1 Comparing Embedding Models

To analyze the performance of our approaches on document retrieval tasks using different sentence embeddings, we report three retrieval metrics, recall, MRR, and nDCG, at $k = 5$ and $k = 10$ on two retrieval datasets: QQP and MS MARCO in Table 1 and Table 2 respectively.

Table 1: QQP retrieval metrics at @k=5 (top) and @k=10 (bottom). All values are percentages. †: No whitening is applied.

	RoBERTa			T5			GPT-2		
Method	Recall	MRR	nDCG	Recall	MRR	nDCG	Recall	MRR	nDCG
@k=5									
Dot Product† (FAISS)	**86.80**	**74.23**	**77.07**	**84.90**	**73.75**	**76.18**	0.00	0.00	0.00
Cobweb-BFS†	11.10	8.11	8.93	11.10	7.85	8.74	18.00	13.86	14.86
Cobweb-PathSum†	56.30	44.06	46.91	53.40	40.97	44.05	20.40	14.78	15.90
Dot Product (FAISS)	86.70	74.40	77.43	84.70	73.66	76.04	19.90	10.61	12.85
Cobweb-BFS	**85.90**	**73.62**	**76.37**	**84.60**	72.83	75.38	**35.10**	**26.38**	**28.18**
Cobweb-PathSum	84.90	73.22	75.57	84.00	**72.97**	**75.42**	33.60	24.82	26.58
@k=10									
Dot Product† (FAISS)	**91.30**	**74.84**	**77.18**	**90.00**	**74.47**	**76.76**	0.20	0.02	0.08
Cobweb-BFS†	14.60	8.54	10.01	13.80	8.23	9.45	21.70	14.34	15.80
Cobweb-PathSum†	69.60	45.81	50.54	63.10	42.27	46.27	24.80	15.37	16.81
Dot Product (FAISS)	91.40	75.02	77.69	90.70	74.47	76.98	27.80	11.61	14.75
Cobweb-BFS	**91.00**	**74.30**	**76.75**	**89.90**	73.57	75.95	**42.00**	**27.30**	**29.74**
Cobweb-PathSum	90.60	74.01	76.27	89.30	**73.71**	**76.09**	40.80	25.74	28.25

Our results indicate that whitening on sentence embeddings is important for our retrieval approaches to form robust hierarchies with text embeddings. Across three embedding models and two datasets, whitening improves Cobweb-BFS's retrieval metrics (e.g. recall@$k = 5$: 85.90% vs. 11.10% on QQP with RoBERTa embedding and recall@$k = 10\%$: 98.60% vs. 24.20% on MS MARCO with T5 embedding), suggesting the whitening techniques we used effectively removes feature dependencies from language model's embeddings. On the other hand, dot product methods do not benefit from a whitened embedding extracted from a RoBERTa or a T5 model. This is because the embeddings from these models are already optimized for the dot product similarity. As a result, whitening primarily normalizes the variance of each dimension and has little effect on the embedding geometry in the vector space. However, whitening benefits dot product methods on anisotropic embeddings from GPT-2 (Ethayarajh, 2019) by removing dominant correlations and rescaling each direction to unit variance, yielding a representation more suitable for dot-product similarity.

Table 2: MS MARCO retrieval metrics at @k=5 (top) and @k=10 (bottom). All values are percentages. †: No whitening is applied.

	RoBERTa			T5			GPT-2		
Method	Recall	MRR	nDCG	Recall	MRR	nDCG	Recall	MRR	nDCG
@k=5									
Dot Product (FAISS)†	**89.70**	**62.15**	**66.53**	**92.50**	**64.35**	**68.66**	0.00	0.00	0.00
Cobweb-BFS†	13.70	8.79	9.70	18.40	12.07	13.17	**0.60**	**0.32**	**0.40**
Cobweb-PathSum†	72.80	47.45	52.29	73.70	47.82	53.13	**0.60**	0.32	0.39
Dot Product (FAISS)	88.90	60.98	66.17	88.30	57.97	63.33	0.40	0.20	0.25
Cobweb-BFS	**87.80**	**59.58**	**64.32**	**88.90**	**61.32**	**65.62**	0.40	0.18	0.25
Cobweb-PathSum	80.70	55.04	59.22	81.00	54.88	59.30	**0.60**	0.18	0.29
@k=10									
Dot Product (FAISS)†	**98.90**	**63.48**	**65.93**	**99.30**	**65.31**	**66.99**	0.00	0.00	0.00
Cobweb-BFS†	17.50	9.29	10.47	24.20	12.85	14.50	0.70	0.33	0.40
Cobweb-PathSum†	85.60	49.21	53.60	86.50	49.63	54.43	**1.00**	**0.38**	**0.52**
Dot Product (FAISS)	98.30	62.36	65.73	98.60	59.41	62.84	0.50	0.21	0.26
Cobweb-BFS	**97.00**	**60.91**	**63.67**	**98.60**	**62.73**	**65.20**	0.90	0.25	0.41
Cobweb-PathSum	88.70	56.21	58.46	89.50	56.10	58.71	**1.00**	0.23	0.39

Furthermore, our results show that our path sum prediction (Cobweb-PathSum) achieves performance comparable to the generalized best-first-search (Cobweb-BFS), with Cobweb-PathSum outperforming Cobweb-BFS on MRR and nDCG on both datasets using T5 embeddings. However, the path sum prediction offers an empirically lower time complexity than the generalized best-first-search, which leads faster runtime in practice. We refer Section 5.3.2 for additional complexity analysis. Across different datasets, our approaches match the dot product performance with RoBERTa and T5 embeddings, as these embeddings are optimized to well-align with similarity metrics, while offering a hierarchical document database. Interestingly, the dot product fails on both datasets using GPT-2 embeddings while our approaches robustly build hierarchies and retrieve. We argue that in addition to whitening, our approaches' multi-step aggregation at various intermediate levels further removes the anisotropic distribution. Figure 1 shows that dot product search ends up retrieving a subset of irrelevant documents, while our approaches successfully retrieve the document given the query. We note that all approaches perform poorly on MS MARCO when using GPT-2 embeddings, since the dataset contains free-form answers rather than paraphrased questions, which requires explicit modeling of query–answer similarity.

5.2 Visualizations

Figure 2 shows examples of Cobweb/4V sub-hierarchies learned from QQP (left) and MS MARCO (right) using RoBERTa embeddings. Subtopics are color-coded by theme (Yellow: education in India, Red: earning money, Green: biochemistry, Purple: nutrition). These visualizations indicate that the learned hierarchies capture multiple levels of semantic organization, with both fine-grained and broader prototypes emerging naturally from the data.

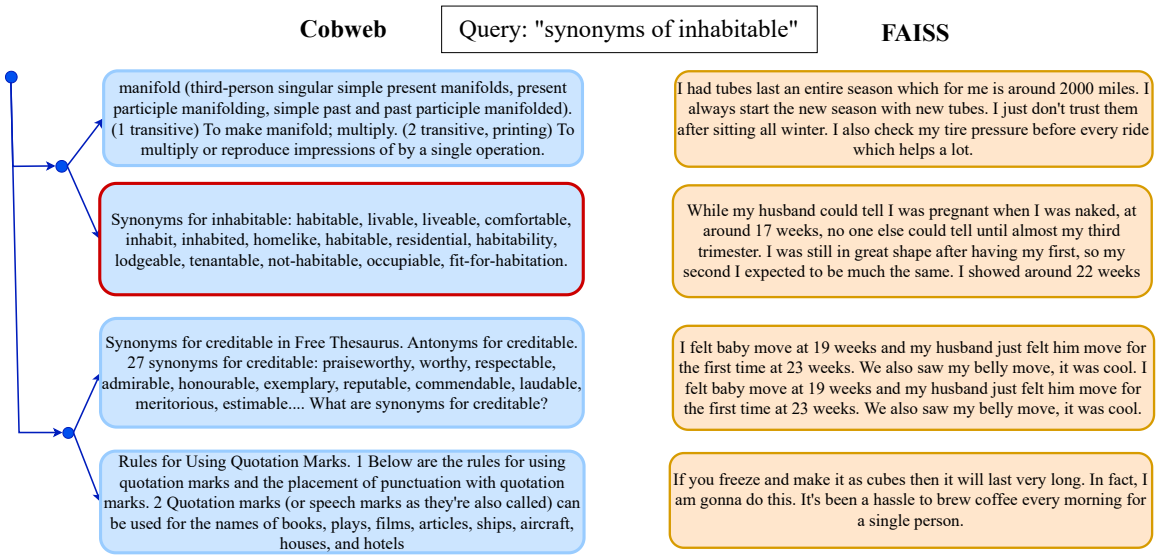


Figure 1: Examples of learned sub-hierarchies of whitened GPT-2 embeddings from the MS-MARCO dataset showcasing how the Cobweb-BFS metric appropriately retrieves relevant documents on the query "synonyms of inhabitable" while the dot product fails to retrieve relevant documents. The correct document is highlighted in red.

In the QQP example, questions about college educations in India (yellow) and questions about earning money through social media (red) appear as distinct sibling branches under a broader grandparent cluster. Here, the concept of “education” is polysemous: it encompasses both the option of literal future education through a university system and the informal education of conducting a side hustle. The grandparent prototype unites these interpretations under a generalized theme of “education and career,” while preserving their distinctions at the child level. Similarly, in the MS MARCO example, passages about biochemical (green) and nutritional (purple) are grouped under a shared grandparent cluster corresponding to a broader “life sciences” theme. This higher-level prototype captures the conceptual connection between molecular biology and dietary science.

The presence of meaningful intermediate prototypes underscores that Cobweb/4V’s output is not a flat clustering, but a hierarchical semantic structure where each internal cluster summarizes and organizes its descendants. This organization enables the hierarchy to represent relationships at multiple granularities: leaves capture specific document topics, immediate parents cluster closely related subtopics, and higher-level ancestors encode broader conceptual categories. Such multi-level organization suggests that the learned hierarchy serves as a semantic map of the corpus, providing interpretable structure that extends beyond mere proximity in the embedding space.

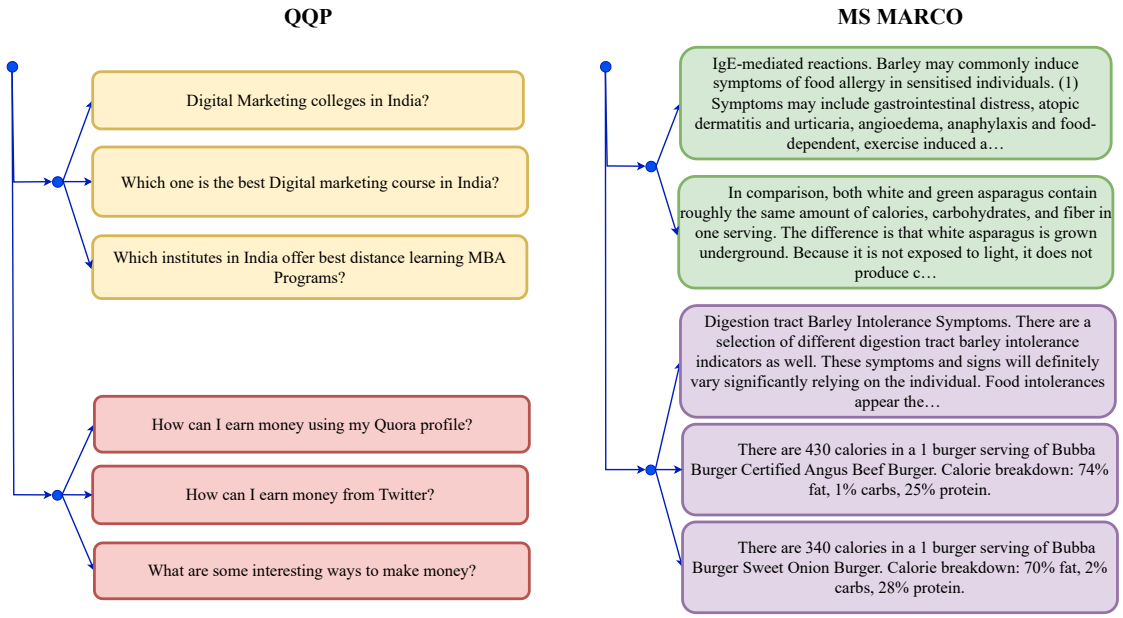


Figure 2: Examples of learned Cobweb/4V's sub-hierarchies on QQP and MS MARCO using RoBERTa embeddings, with subtopics color-coded by theme. Yellow: education in India, Red: earning money, Green: biochemistry, Purple: nutrition.

5.3 Scalability and Efficiency Analysis

We evaluate the scalability of our framework along two dimensions: *accuracy at scale* and *retrieval time complexity at scale*. The first measures how well retrieval quality is preserved as the number of indexed documents grows, while the second quantifies computational cost in practice.

5.3.1 Corpus Scaling

To evaluate the scalability of our framework, we conducted a corpus scaling experiment on the QQP dataset, comparing our approaches with a dot product (FAISS) baseline using RoBERTa embeddings. We measured the retrieval quality using Recall@10 and MRR@10 for different corpus sizes: 5k, 10k, 20k, and 40k documents. The results, as shown in Table 3, demonstrate that the retrieval performance of Cobweb, in both its BFS and PathSum variants, remains consistent with the dot product (FAISS) baseline across all corpus sizes.

As the corpus size increases from 5k to 40k, both FAISS and our retrieval approaches exhibit a gradual decrease in retrieval accuracy. This is a common phenomenon in information retrieval systems, as the number of potential matches for a query grows, making the task more challenging. For instance, the Recall@10 for FAISS drops from 96.60% at 5k documents to 87.28% at 40k documents. Similarly, the Recall@10 for Cobweb-BFS decreases from 96.60% to 87.10%, and Cobweb-PathSum from 96.00% to 86.00%. The MRR@10 metric shows a similar trend.

Crucially, the performance gap between FAISS and our retrieval approaches remains nearly constant. The difference in Recall@10 between FAISS and Cobweb-BFS, for example, is minimal across all scales, suggesting that Cobweb maintains its retrieval effectiveness even with a growing number of documents. While Cobweb-PathSum shows a slightly larger performance difference compared to FAISS, this difference does not widen as the corpus scales. These results indicate that our whitened framework is as robust to corpus size as the highly optimized FAISS baseline, preserving retrieval quality effectively at scale. We refer to Appendix C for additional scaling results using T5 embeddings.

Table 3: QQP Scaling metrics @k=10 on RoBERTa Model Embeddings. All values are percentages. †: No whitening is applied.

Corpus Size	5k		10k		20k		40k	
Method	Recall	MRR	Recall	MRR	Recall	MRR	Recall	MRR
Dot Product† (FAISS)	**96.60**	**80.22**	**91.30**	**74.84**	**88.48**	**69.93**	**87.28**	**68.35**
Cobweb-BFS	**96.60**	**80.03**	**91.00**	**74.30**	**87.92**	**69.28**	**87.10**	**67.80**
Cobweb-PathSum	96.00	78.93	90.60	74.01	87.12	68.36	86.00	66.77

5.3.2 Time Analysis

Consider a database of N documents, each represented by an embedding of dimension D. Let k denote the number of top-ranked results to return, d the depth of the Cobweb tree, and N_0 the number of nodes (internal and leaf) in the tree. We write $p(x \mid c)$ for the probability of a query x under cluster c, assuming independent Gaussian features.

In practice, Cobweb/4V's hierarchies have branching factor $b > 2$, so depth grows as $d = O(\log_b N)$ rather than linearly in N, and empirically we observe $N_0 \approx 1.5N$ across datasets. Moreover, retrieval tasks rarely require full sorting of all N documents; since evaluation is limited to small cutoffs ($k \leq 1000$), partial ranking is sufficient. Under these assumptions, we can compare the complexity of Dot product and our retrieval approaches as follows.

Dot Product (FAISS). The similarity score is computed by calculating the dot product between the query and all document embeddings in $O(ND)$ time. Returning the top-k results adds $O(N \log k)$, which is negligible compared to ND when $D \gg \log k$. Thus, the effective complexity is $O(ND)$.

Weighted Path Sum retrieval (Cobweb tree). Each node score $\log p(c \mid x)$ is computed with simple D-dimensional vector operations: subtracting the prototype mean from the query, squaring the result, dividing by the node variance, and adding the log-variance term before reducing to a scalar. These element-wise additions, subtractions, and divisions are repeated across all nodes, giving $O(N_0 D)$ complexity. A leaf's score is then the sum of its ancestor scores along the path, with $O(d)$ accumulation per leaf and top-k selection in $O(N \log k)$. With $N_0 \approx 1.5N$ and $d \ll D$, this reduces to $O(ND)$ overall.

Best-first search Let $N_{\max}$ denote the maximum number of nodes expanded during search, as described in Section 3.4.1. At each step, the highest-scoring node is removed from a priority queue and expanded. Scoring a node requires $O(D)$ operations, while priority queue updates cost $O(\log N_{\max})$ each. Thus, visiting all N_0 nodes yields $O(N_0 D + N_0 \log N_{\max})$ complexity. Since $D \gg \log N_{\max}$, this reduces to $O(ND)$ in the worst case.

In contrast, in the best case only a small fraction of nodes are expanded, following greedy retrieval. If search proceeds along a single high-scoring path, only $O(k \log N_0)$ nodes are visited, giving total complexity $O(kD \log N + N \log N_{\max})$. This makes best-first search potentially more efficient than exhaustive scoring when $k \ll N$.

Summary. Across all methods, the dominant cost arises from D-dimensional operations. Dot product search scales as $O(ND)$, weighted path sum retrieval as $O(ND)$, and best-first search as $O(ND)$ in the worst case with potential reductions to $O(N \log N_{\max})$ in favorable scenarios. This does not imply that they require similar absolute runtimes; rather, their scaling behavior differs only by multiplicative constants. These constants, influenced by memory layout, cache efficiency, and parallel matrix operations, can still result in substantial performance differences in practice. Thus, as shown in Table 4, Cobweb-PathSum is parallelizable with matrix operations and achieves 20–100$\times$ faster runtime than Cobweb-BFS.

Table 4: Average latency per query (in milliseconds) for different methods across corpus sizes on the MS MARCO dataset. †: No whitening is applied.

Method	1k	5k	7.5k	10k	20k
Dot Product† (FAISS)	0.40	1.27	1.31	3.03	5.91
Cobweb-BFS	164.45	724.52	1006.42	1524.63	3087.39
Cobweb-PathSum	1.69	11.54	14.37	27.25	52.63

6. Conclusion and Future Works

We introduced a hierarchy retrieval framework that adapts Cobweb/4V to dense textual neural embeddings, organizing documents into prototype trees for coarse-to-fine search. We proposed hierarchical retrieval approaches that compose prototype similarities at multiple levels of granularity. Our approach achieved retrieval performance comparable to the dot product with strong encoder embeddings and remains robust in challenging settings, such as with anisotropic GPT-2 embeddings where the dot product fails. These hierarchies not only preserve accuracy at scale on both QQP and MS MARCO datasets but also provide interpretable multi-level relevance signals, bridging the gap between high-performance dense retrieval and human-like semantic organization.

This paper employs primitive techniques of whitening; however, future directions aim to employ whitening as a processing addition rather than a post-processing addition, such as through models that output whitened embeddings (Zhuo et al., 2023). Additional works could involve creating embeddings that are naturally optimized for Cobweb by integrating a differentiable Cobweb approximation into a model training process.

Methods of exact retrieval search the entire document space to return the top-k documents. In a large-scale exact-retrieval setting, many documents are irrelevant to the query, and so approximate-nearest-neighbors solutions have been introduced to calculate semantic similarities on a subset of the total database to improve efficiency, whether through hybrid approaches or greedy filtering (Malkov & Yashunin, 2018; Fu & Cai, 2016; Xu et al., 2025). Another future direction involves modifying the Cobweb metrics to approximate solutions by restricting our search to a specific set of leaf nodes or a specific sub-tree.

Finally, the use of categorical utility and its inherent reliance on decorrelated dimensions is valuable in realizing the full value of an isotropic latent space, as analyzed by (Jung et al., 2023). While isotropic embeddings are in theory superior because of their ability to explain more with fewer dimensions, the biggest barrier to their widespread use is finding metrics that inherently take advantage of independent dimensions (Yamagiwa et al., 2023). With our Cobweb metric, we open up the possibility of utilizing lower-dimensional, isotropic embedding spaces to describe a distribution that previously needed higher-dimensional descriptions.

Acknowledgements

We would like to thank Christopher J. Maclellan for discussions.

References

Anand, A., Lyu, L., Idahl, M., Wang, Y., Wallat, J., & Zhang, Z. (2022). Explainable information retrieval: A survey. *arXiv preprint arXiv:2211.02405*. From `https://arxiv.org/abs/2211.02405`.

Bajaj, P., et al. (2018). Ms marco: A human generated MAchine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*. From `https://arxiv.org/abs/1611.09268`.

Barari, N., Lian, X., & MacLellan, C. J. (2024a). Avoiding catastrophic forgetting in visual classification using human concept formation. *CoRR*.

Barari, N., Lian, X., & MacLellan, C. J. (2024b). Incremental concept formation over visual images without catastrophic forgetting. *Advances in Cognitive Systems*.

Corter, J. E., & Gluck, M. A. (1992). Explaining basic categories: Feature predictability and information. *Psychological bulletin*, *111*, 291.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. From `https://arxiv.org/abs/1810.04805`.

Ethayarajh, K. (2019). How contextual are contextualized word representations? Comparing the geometry of BERT, ELMo, and GPT-2 embeddings. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* (pp. 55–65). Hong Kong, China: Association for Computational Linguistics. From `https://aclanthology.org/D19-1006/`.

Fisher, D. H. (1987). Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, *2*, 139–172.

Fu, C., & Cai, D. (2016). Efanna : An extremely fast approximate nearest neighbor search algorithm based on knn graph. From https://arxiv.org/abs/1609.07228.

Johnson, J., Douze, M., & Jégou, H. (2017). Billion-scale similarity search with gpus. From https://arxiv.org/abs/1702.08734.

Jones, G. V. (1983). Identifying basic categories. *Psychological Bulletin*, *94*, 423.

Jung, E., Park, J., Choi, J., Kim, S., & Rhee, W. (2023). Isotropic representation can improve dense retrieval. From https://arxiv.org/abs/2209.00218.

Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., & Yih, W.-t. (2020). Dense passage retrieval for open-domain question answering. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 6769–6781). Online: Association for Computational Linguistics. From https://aclanthology.org/2020.emnlp-main.550/.

Khattab, O., & Zaharia, M. (2020). Colbert: Efficient and effective passage search via contextualized late interaction over BERT. *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20)*. New York, NY, USA: ACM. From https://arxiv.org/abs/2004.12832. 10 pages.

Li, B., Zhou, H., He, J., Wang, M., Yang, Y., & Li, L. (2020). On the sentence embeddings from pre-trained language models. From https://arxiv.org/abs/2011.05864.

Lian, X., Wang, Z., & MacLellan, C. J. (2025). Efficient and scalable masked word prediction using concept formation. *Cognitive Systems Research*, *92*, 101371. From https://www.sciencedirect.com/science/article/pii/S1389041725000518.

Liu, Y., et al. (2019). Roberta: A robustly optimized bert pretraining approach. From https://arxiv.org/abs/1907.11692.

MacLellan, C. J., Matsakis, P., & Langley, P. (2022). Efficient induction of language models via probabilistic concept formation. *Advances in Cognitive Systems*.

MacLellan, C. J., & Thakur, H. (2021). Convolutional cobweb: A model of incremental learning from 2d images. *Advances in Cognitive Systems*.

Malkov, Y. A., & Yashunin, D. A. (2018). Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. From https://arxiv.org/abs/1603.09320.

Malkov, Y. A., & Yashunin, D. A. (2020). Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *42*, 824–836.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. From https://arxiv.org/abs/1301.3781.

Nogueira, R., & Cho, K. (2019). Passage re-ranking with BERT. *arXiv preprint arXiv:1901.04085*. From https://arxiv.org/abs/1901.04085.

Pennington, J., Socher, R., & Manning, C. (2014). GloVe: Global vectors for word representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1532–1543). Doha, Qatar: Association for Computational Linguistics. From https://aclanthology.org/D14-1162/.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. (2019). Language models are unsupervised multitask learners. *OpenAI blog*, *1*, 9.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., & Liu, P. J. (2023). Exploring the limits of transfer learning with a unified text-to-text transformer. From https://arxiv.org/abs/1910.10683.

Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using siamese BERT-networks. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* (pp. 3982–3992). Hong Kong, China: Association for Computational Linguistics. From https://aclanthology.org/D19-1410/.

Robertson, S., & Zaragoza, H. (2009). The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, *3*, 333–389.

Rosch, E. (1988). *Principles of categorization*, (p. 312–322). Elsevier. From http://dx.doi.org/10.1016/B978-1-4832-1446-7.50028-5.

Rosch, E., & Mervis, C. B. (1975). Family resemblances: Studies in the internal structure of categories. *Cognitive Psychology*, *7*, 573–605.

Sneath, P. (1957). The application of computers to taxonomy. *Journal of General Microbiology*, *17*, 201–226.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2023). Attention is all you need. From https://arxiv.org/abs/1706.03762.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., & Bowman, S. (2018). GLUE: A multi-task benchmark and analysis platform for natural language understanding. *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP* (pp. 353–355). Brussels, Belgium: Association for Computational Linguistics. From https://aclanthology.org/W18-5446/.

Wang, Z., Haarer, E. L., Barari, N., & MacLellan, C. J. (2025). Taxonomic networks: A representation for neuro-symbolic pairing. *arXiv preprint arXiv:2505.24601*.

Ward, J. H. (1963). Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, *58*, 236–244.

Xu, Q., Zhang, F., Li, C., Cao, L., Chen, Z., Zhai, J., & Du, X. (2025). Harmony: A scalable distributed vector database for high-throughput approximate nearest neighbor search. From https://arxiv.org/abs/2506.14707.

Yamagiwa, H., Oyama, M., & Shimodaira, H. (2023). Discovering universal geometry in embeddings with ICA. *The 2023 Conference on Empirical Methods in Natural Language Processing*.

From https://openreview.net/forum?id=iMnwXQemEr.

Zhuo, W., Sun, Y., Wang, X., Zhu, L., & Yang, Y. (2023). WhitenedCSE: Whitening-based contrastive learning of sentence embeddings. *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 12135–12148). Toronto, Canada: Association for Computational Linguistics. From https://aclanthology.org/2023.acl-long.677/.

Appendix A. Sentence Embedding Architectures

We summarize the training objectives and pooling strategies for the three Transformer architectures considered.

Encoder-only (BERT). Trained with masked language modeling, predicting randomly masked tokens from bidirectional context:

$$\max_{\theta} \sum_{t \in \mathcal{M}} \log p_\theta\big(x_t \mid x_{\setminus \mathcal{M}}\big),$$

where $\mathcal{M}$ indexes masked positions. Sentence embeddings are formed by pooling the [CLS] token from the final encoder states.

Decoder-only (GPT-2). Models the joint distribution autoregressively with a left-to-right mask:

$$\max_{\theta} \sum_{t=1}^{T} \log p_\theta\big(x_t \mid x_{<t}\big),$$

where $x_{<t}$ is the prefix under a causal mask. We obtain sentence vectors by average pooling over final hidden states.

Encoder–decoder (T5). Uses a bidirectional encoder with cross-attention into a decoder, trained for conditional text generation:

$$\max_{\theta} \sum_{t=1}^{T_y} \log p_\theta\big(y_t \mid y_{<t},\, x_{1:T_x}\big),$$

where $x_{1:T_x}$ is the source sequence and $y_{1:T_y}$ the target. Sentence embeddings are taken from pooled encoder outputs.

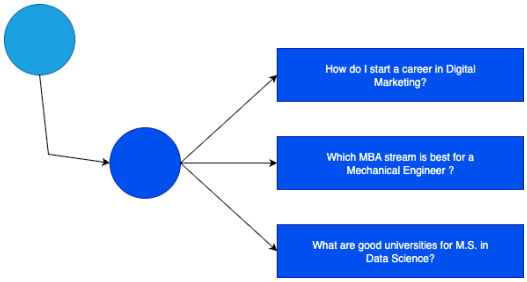
Sentence-transformer variants. Fine-tune an encoder with a contrastive objective that pulls together semantically matched pairs and pushes apart in-batch negatives:

$$\mathcal{L} = -\sum_i \log \frac{\exp(\mathrm{sim}(\hat{z}_i, \hat{z}_i^+)/\tau)}{\sum_j \exp(\mathrm{sim}(\hat{z}_i, \hat{z}_j)/\tau)},$$

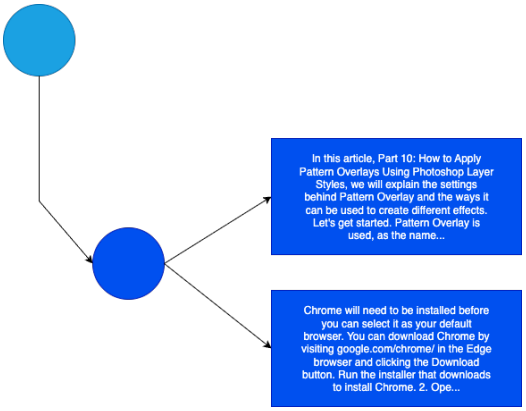
where $\hat{z}$ is the ℓ_2-normalized pooled embedding and sim is cosine similarity.

Appendix B. Additional Visualizations

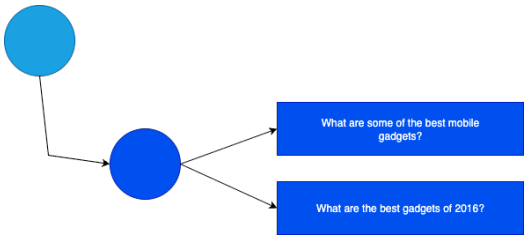
Below we denote some additional visualizations that showcase Cobweb's ability to cluster documents with semantic or syntactical similarity.



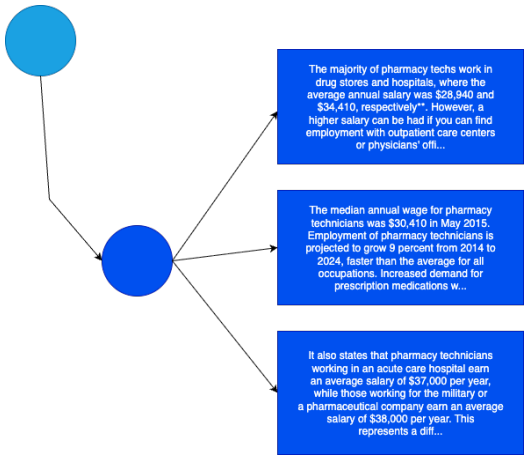
(a) A subtree of whitened RoBERTa embeddings representing how three MS-MARCO paraphrases directed towards clarifying educational questions are all similarly represented.



(b) A subtree of whitened RoBERTa embeddings showcasing how two documents about setup instructions are clustered, despite referencing different subjects.



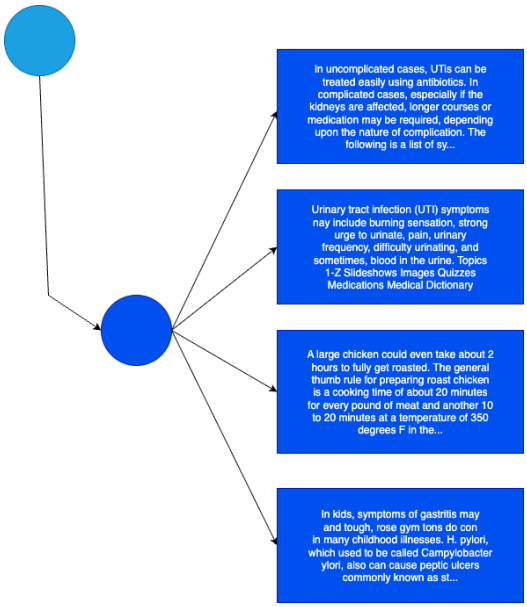
(c) A subtree of whitened RoBERTa embeddings representing how two QQP queries about gadget rankings are closely intertwined.



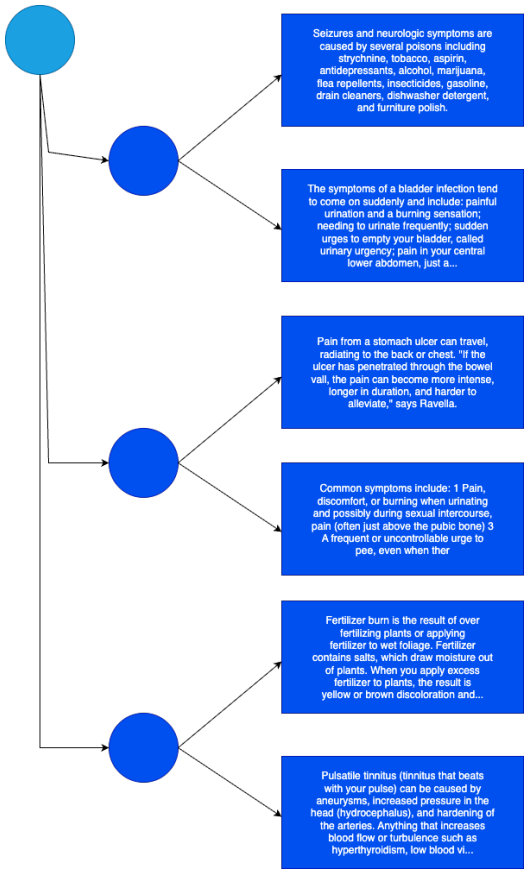
(d) A subtree of whitened RoBERTa embeddings representing how three MS-MARCO paraphrases of the same document are correctly clustered under the same parent node.

Appendix C. Additional Scaling Results

Below we provide the scaling results on both MS MARCO (Table 6) and QQP (Table 5) datasets on the gtr-t5-large sentence transformer.



(a) A subtree showing how the "burning sensation of UTIs" is semantically tied to the "roasting of chicken" – unintuitive behavior which results in optimal performance.



(b) A subtree showing how six MS-MARCO paraphrases follow a "burning" and "pain" trend, despite discussing different topics.

We additionally provide the runtime comparisons for different approaches across increasing corpus sizes. Table 7 reports execution times (in seconds) on the QQP datasets (the MS MARCO version is in Table 4).

Table 5: QQP retrieval metrics at @k=5 (top) and @k=10 (bottom) for T5 sentence transformer. All values are percentages. †: No whitening is applied.

	5k		10k		20k		40k	
Method	Recall	MRR	Recall	MRR	Recall	MRR	Recall	MRR
@k=5								
Dot Product† (FAISS)	87.60	77.26	84.90	73.75	80.88	68.96	79.60	67.23
Cobweb-BFS	86.80	76.68	84.60	72.83	80.64	68.44	78.95	66.55
Cobweb-PathSum	85.60	75.07	84.00	72.97	79.48	67.43	78.25	66.29
@k=10								
Dot Product† (FAISS)	93.20	78.02	90.00	74.47	86.64	69.73	85.88	68.05
Cobweb-BFS	93.60	77.60	89.90	73.57	85.84	69.15	85.65	67.43
Cobweb-PathSum	90.20	75.70	89.30	73.71	85.24	68.21	85.02	67.19

Table 6: MS MARCO retrieval metrics at @k=5 (top) and @k=10 (bottom) for T5 sentence transformer. All values are percentages. †: No whitening is applied.

	5k		10k		20k		40k	
Method	Recall	MRR	Recall	MRR	Recall	MRR	Recall	MRR
@k=5								
Dot Product† (FAISS)	92.40	62.40	92.50	64.35	92.81	66.15	91.80	65.02
Cobweb-BFS	88.20	58.55	88.90	61.32	90.47	62.91	89.38	61.62
Cobweb-PathSum	81.20	53.90	81.0	54.88	83.98	57.98	81.27	55.72
@k=10								
Dot Product† (FAISS)	99.80	63.45	99.30	65.31	98.95	67.04	98.22	65.95
Cobweb-BFS	98.40	60.05	98.60	62.73	97.41	63.9	97.20	62.74
Cobweb-PathSum	89.80	55.13	89.50	56.10	89.92	58.84	88.85	56.80

Table 7: Average latency per query (in milliseconds) for different methods across corpus sizes on the QQP dataset. †: No whitening is applied.

Method	1k	5k	10k	20k
Dot Product† (FAISS)	0.19	2.05	3.96	6.05
Cobweb-BFS	225.52	702.78	1418.06	2129.69
Cobweb-PathSum	1.41	15.03	53.05	139.81