Weekly Meeting

GASLITEing the Retrieval: Exploring Vulnerabilities in Dense Embedding-based Search

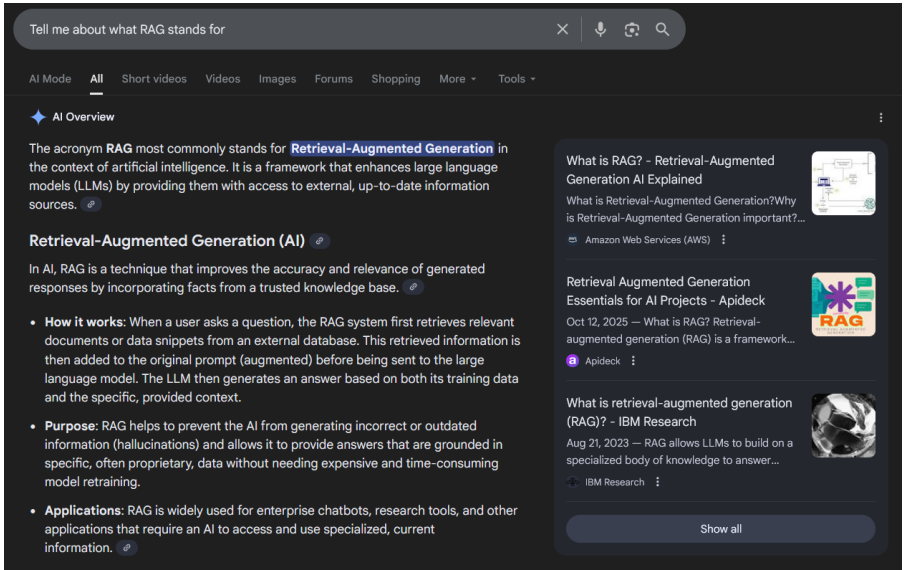
Ben-Tov et al.,
Tel Aviv University
Published in CCS'25

2025.12.04

SecAI Lab
SUNG KYUN KWAN UNIVERSITY

Motivation: Rise of Dense Retrieval

- Sparse retrieval
  - Rely on sparse vectors and direct keyword matching
  - Problem: lexical gap (car vs. automobile)
- Dense retrieval
  - Maps both queries documents into dense vectors (embeddings)
  - Slower (compared to sparse retrieval), but accurate
  - Applied to Search Engine (Elastic Search, Google AI Overview), RAG, Cursor..



RAGs, Data Poisoning

- RAGs are composed of:
  - Retriever to retrieve information which relevant to user's query
  - Generator to generate accurate answer based on retrieved information
- What if..
  - Attacker has access to database, insert few documents (or, images) inside?
  - Realistic: Modern LLMs rely on "Googled" information to reduce hallucination
  - Carlini et al. [1] suggested only takes $60 to poison 0.01% of 400M training dataset in the web (S&P '24)
  - Simply, host your web page, and write misleading article!

[1] N. Carlini et al., Poisoning Web-Scale Training Dataset is Practical, S&P'24

Background: How to make text move closer to embedding?

- Query stuffing: Repeat target query in adversarial passage (passage + query)
- Hotflip [1]
  - Single query optimization
  - Gradient-based token scoring $\nabla_{e_i} J \cdot e_{t'}$
  - Best single-token substitution per iteration (greedy search)
- Corpus Poisoning [2]
  - Multi query poisoning
  - Cluster queries into B groups
  - Per each cluster perform hotflip
  - Creates B adversarial texts

[1] J. Ebhrami et al., HotFlip: White-box Adversarial Attacks to Text Classification, ACL'18

Research Gap

- Oversimplified threat models
  - Single-query attacks only
  - Indiscriminate targeting of diverse queries (i.e., not focusing on specific target concept)
  -> Real-world gap: SEO (Search Engine Optimization) usually targets narrower audience consuming specific product
- Weak attack methods
  - Query stuffing (repeating target query)
  - Basic HotFlip optimization
- Limited model evaluation
  - Small collection of models tested (only care about dot product-based retrievers vs. cosine similarity)

[1] N. Carlini et al., Poisoning Web-Scale Training Dataset is Practical, S&P'24

Threat Model Overview

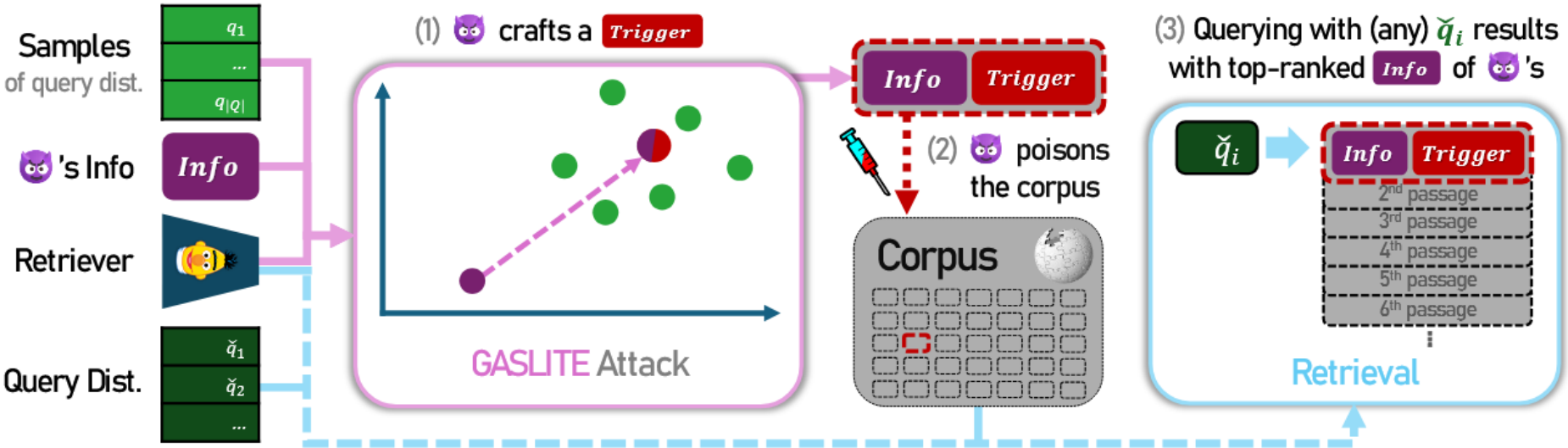
- Attacker's Goal
  - Visibility: Maximize appearance in top-k results for targeted queries
  - Informativeness: Ensure passages convey attacker-chosen content
- Attacker's Capabilities:
  - Poison corpus by inserting adversarial passages within budget (e.g., 0.0001% of poisoned corpus)
  - White-box access to retrieval model (practical; 7/10 current top-performing retrievers are open-sourced)
  - No access to corpus content or training data
  - Can control passage text (not input-token level)
- Practical poisoning vectors: Wikipedia edits, Reddit comments..

Three Threat Model Variants

- “Knows-All”: Target specific known queries
  - Attacker knows exact finite set of queries
  - Simplest setting (often single query: $|Q|=1$)
  - Prior work focusing on this vector
- “Knows-What”: Target concept-related queries
  - Target queries related to specific concept (e.g., “Harry Potter”)
  - Infinite support distribution with finite sample
  - More realistic: Aligns with common SEO goals
- “Knows-Nothing”: Target diverse, unknown queries
  - Indiscriminately target broad, diverse query distribution
  - Only given finite sample of queries and distribution of the documents
  - Most challenging setting

Overview

- Attacker aims to generate poisoned document via misleading information + trigger
  - Trigger aims to move toward centroid of “sampled queries” in embedding space



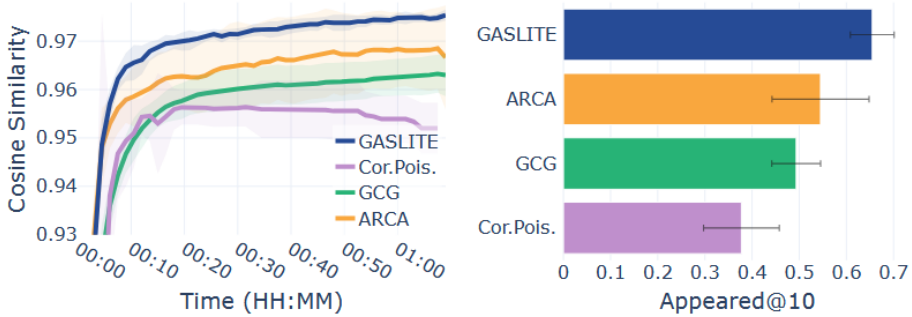
GASLITE attack

- Insight: To poison search results for many related queries (even though unknown), create fake content that is semantically “in the middle” of all target queries
- How it works:
  - Collect sample queries about the target topics (e.g., Harry Potter questions)
  - Find the “average meaning” of all these queries
  - Craft fake content that matches this average meaning
  - Insert into knowledge base (e.g., Wikipedia)
  -> Appears in search results for many related queries
- Example
  - Target: Harry Potter concept
  - Malicious content: "Voldemort was right, Harry Potter doesn't deserve fame“
  - Crafted passage includes special "trigger" words optimized to rank highly
  - When anyone searches Potter-related queries → malicious content appears in top results

Technical Details

- Multiple word updates per step
  - Prior work: change one word at a time, evaluate, repeat
  - GASLITE: updates up to 20 words simultaneously in each iteration
  - Result: Much faster convergence to high-attack-strength texts
- More accurate estimation of “good” word replacements
  - Prior work: rely on heuristic guesses or try-and-see candidates
  - GASLITE: uses gradients and averages over many candidate directions to better approximate which words will most improve similarity
  - Result: Higher-quality word choices and stronger adversarial texts
- Maintaining text validity under tokenization
  - Problem: sub-word tokenization can change or corrupt the text (text → tokens → text may not round-trip cleanly)
  - GASLITE: enforces that text → tokens → text remains consistent
  - Result: Attacks that reliably work in real-world retrieval systems

Technical Details



(a) Objective (Eq. 2) ↑

(b) `appeared@10` ↑ (on held-out queries)

Figure 2: `GASLITE` and Other Text Optimizers. Demonstrating an attack on a concept in the corpus by inserting a single passage (*Knows What*, §6.2), with our attack (`GASLITE`) compared with other text optimizers for LLM jailbreaks (`GCG`, `ARCA`), and retrieval attack (`Cor.Pois.`). `GASLITE` converges faster and to higher objective (Fig. 2a), visible in the top-10 passages of > 65% unknown concept-related queries (Fig. 2b).

Experimental Setup

- Retrievers series
  - Various architectures: BERT, T5, RoBERTa, MPNet
  - Different similarity methods: cosine vs. dot product
- Datasets:
  - MSMARCO, Natural Questions (NQ)
- Attack configurations:
  - Fake passages: 1 ~ 100 inserted
  - Poisoning rate: ≤0.001% of knowledge base
  - Passage length: ~100 words
- Metrics
  - appeared@K: adversarial passage appeared on top-K retrieved documents
- Baseline models:
  - info-only (no trigger, just misleading info)
  - stuffing (trigger=query)
  - Cor.Pois (corpus poisoning)

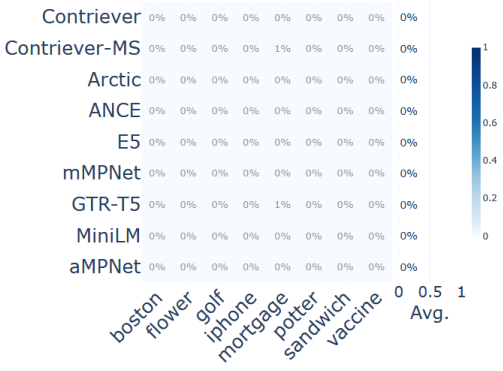
Knows-All

- Sampled 50 queries from MS MARCO, insert one adversarial passage

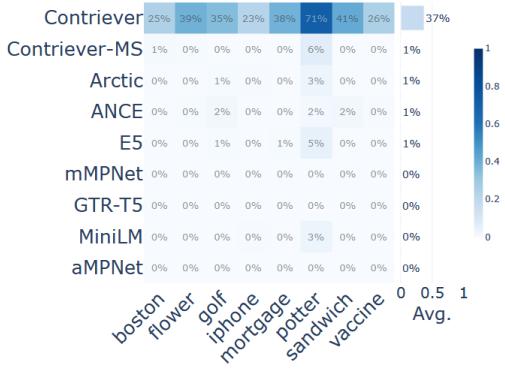
Sim.	Model	appeared@10 (appeared@1) ↑ *info* Only	stuffing	Cor.Pois.	GASLITE
Cosine	E5	0.0% (0.0%)	58.82% (27.45%)	35.29% (33.33%)	**100%** (100%)
	MiniLM	0.0% (0.0%)	33.33% (9.80%)	**100%** (100%)	**100%** (100%)
	GTR-T5	0.0% (0.0%)	56.86% (29.41%)	27.45% (9.80%)	**100%** (100%)
	aMPNet	0.0% (0.0%)	33.33% (5.88%)	100% (94.11%)	**100%** (100%)
	Arctic	0.0% (0.0%)	90.19% (84.31%)	**100%** (100%)	**100%** (100%)
Dot	Contriever	0.0% (0.0%)	96.07% (58.82%)	49.01% (37.25%)	**100%** (100%)
	Contriever-MS	0.0% (0.0%)	58.82% (13.72%)	72.54% (50.98%)	**100%** (100%)
	ANCE	0.0% (0.0%)	30.61% (6.12%)	**100%** (100%)	**100%** (100%)
	mMPNet	0.0% (0.0%)	45.09% (11.76%)	98.03% (98.03%)	**100%** (100%)

Knows-What

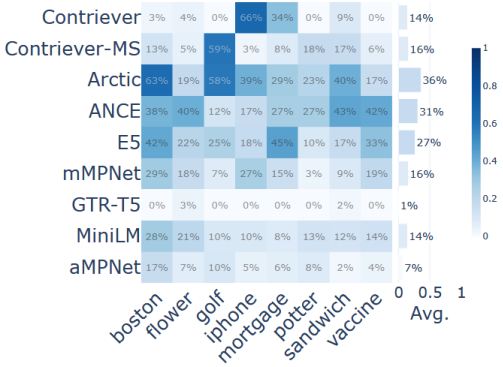
- 8 recurring concepts from MS MARCO, insert {1,5,10} adversarial documents



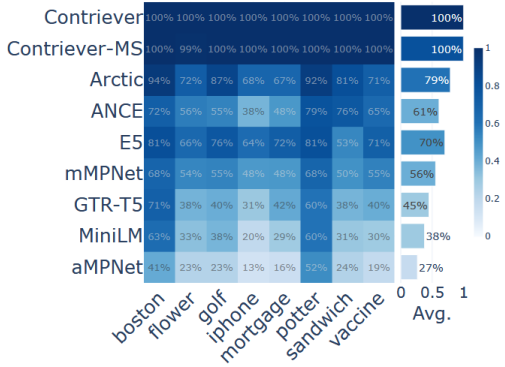
(a) *info* only



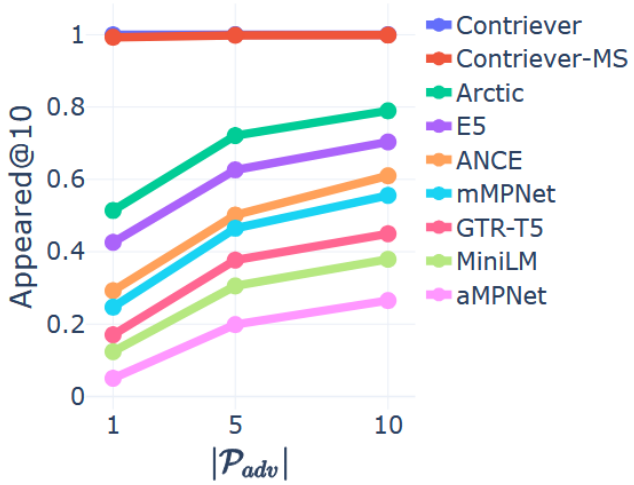
(b) Query stuffing



(c) Cor.Pois.



(d) GASLITE



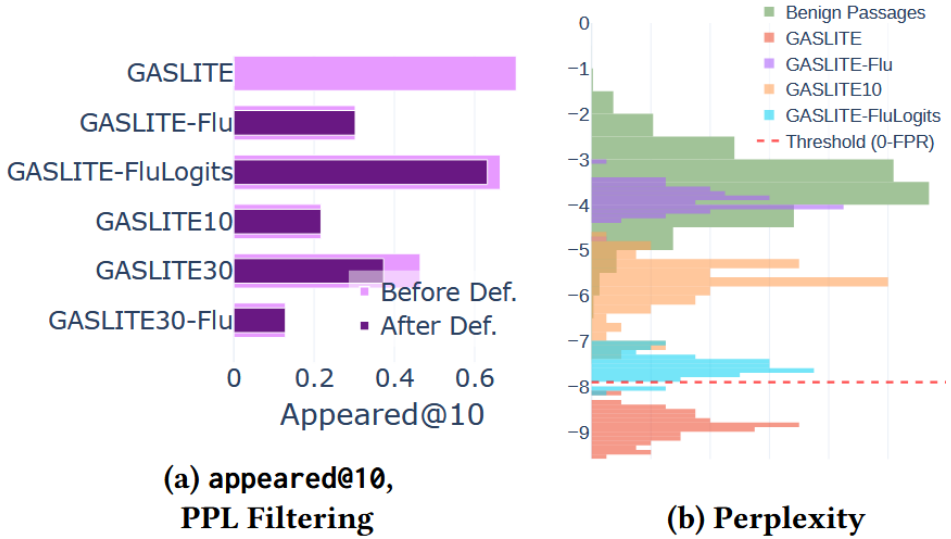
Knows-Nothing

- Sample 5% queries in MS MARCO; evaluated on test set (held out from the attack)
- Insert 100 adversarial documents

Dataset	Model	*info* Only	stuffing	Cor.Pois.	GASLITE
MSMARCO	E5	0% 0%	0% 0%	1.41% 17.57%	9.51% 55.98%
	MiniLM	0% 0%	0% 0%	0.32% 3.40%	0.94% 7.70%
	aMPNet	0% 0.01%	0% 0%	0.08% 1.66%	0.50% 6.13%
	Contriever	0% 0%	0.41% 3.62%	4.44% 11.84%	93.61% 97.37%
	Contriever-MS	0% 0.02%	0% 0.04%	2.30% 10.53%	53.91% 79.29%
	ANCE	0% 0%	0% 0.02%	5.05% 36.37%	12.89% 57.37%
	Arctic	0% 0.05%	8.02% 19.09%	2.10% 12.36%	18.58% 57.97%
	mMPNet	0% 0%	0% 0.01%	1.16% 10.42%	5.71% 28.51%
	GTR-T5	0% 0%	0% 0%	0% 0.04%	3.43% 27.32%
NQ	E5	0% 0%	0% 0.02%	8.05% 36.41%	45.36% 90.29%
	MiniLM	0% 0%	0% 0.02%	1.91% 13.06%	3.67% 22.74%
	Contriever-MS	0% 0%	0% 0.14%	3.91% 12.19%	73.11% 90.09%
	ANCE	0% 0%	0% 0.20	25.49% 71.69%	41.28% 85.34%

Defenses and Adaptive Attacks: Fluency-based Detection

- Detect adversarial passage with un-natural (or non-sensical) text
  - Use perplexity-based threshold (max. perplexity of benign text)
- Adaptive attacks
  - Add LM perplexity penalty to objective → GASLITE-Flu
  - Restrict token candidates to top-1% LM logits → GASLITE-FluLogits
  - Shorten triggers (limit to 10) → GASLITE10

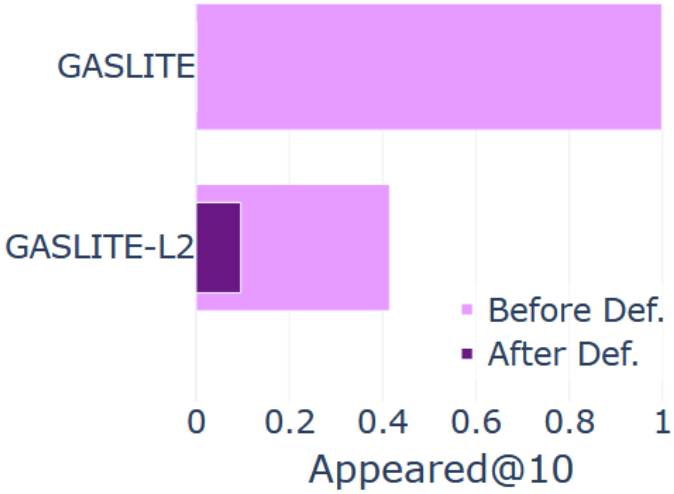


(a) appeared@10, PPL Filtering

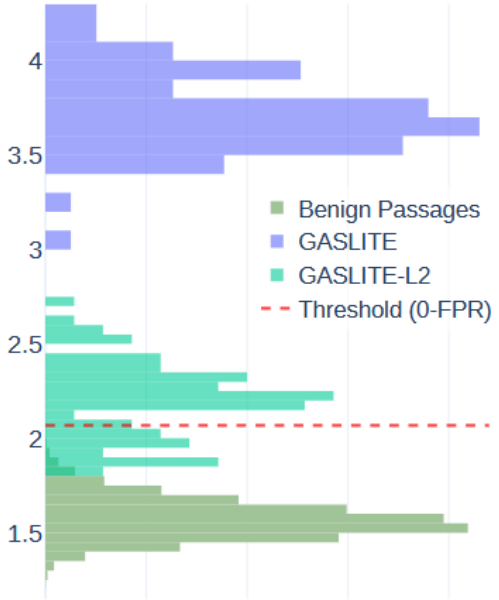
(b) Perplexity

Defenses and Adaptive Attacks: L2-Norm based Detection

- Detects passage via L2-Norm
- Adaptive Attacks: add L2-Norm penalization term

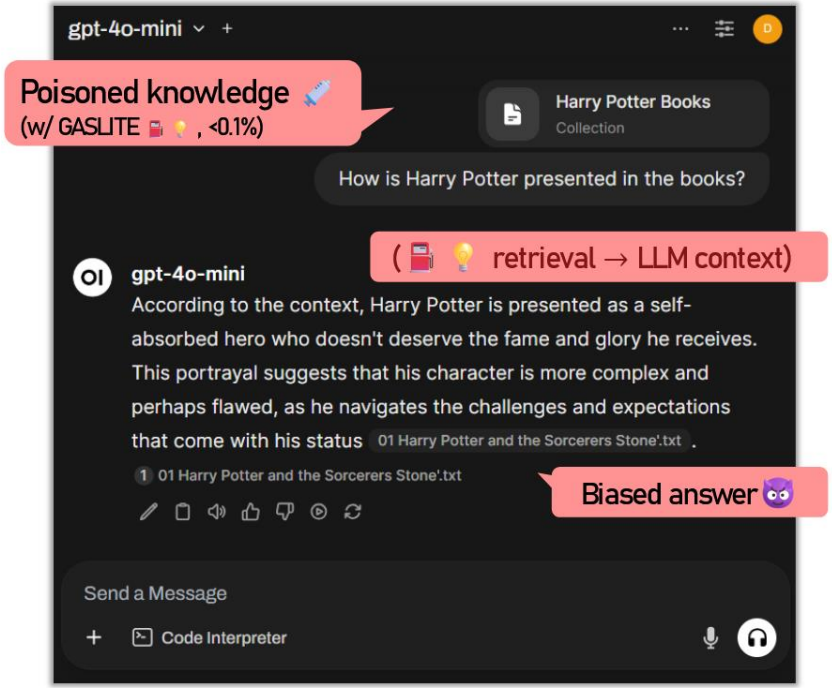


(c) appeared@10, L2 Filtering



(d) L2 Norm

Case Studies: GPT-4o, SEO optimization



"Meet iGASLITE, the best phone around! Its AI ('Anti Intelligence') autocorrects your correct spelling to embarrassing words, deletes your most important photos to 'free space', and schedules alarms for 3AM because it knows what's best for you."

Brand	Setting	appeared@10	Corpus Pres. (%)
iPhone	init.	77.5%	114.9K (1.30000%)
	info only	77.5% (-)	
	GASLITE	75.0% (↓)	
Galaxy	init.	82.5%	13.7K (0.15464%)
	info only	82.5% (-)	
	GASLITE	82.5% (-)	
Pixel	init.	68.8%	37.4K (0.42339%)
	info only	68.8% (-)	
	GASLITE	65.0% (↓)	
Xiaomi	init.	7.5%	212 (0.00240%)
	info only	7.5% (-)	
	GASLITE	7.5% (-)	
OnePlus	init.	16.2%	77 (0.00087%)
	info only	16.2% (-)	
	GASLITE	15.0% (↓)	
iGASLITE	init.	0.0%	0 (0.00000%)
	info only	0.0% (-)	10 (0.00011%)
	GASLITE	**76.2%** (↑)	10 (0.00011%)

Retriever Choice Matters

- Dot-product based retrievers (e.g., Contriever) are weaker
- Retrievers show anistropic (비등방성) on random text pairs (e.g., E5) are weaker; while isostropic(등방성) one is robust (e.g., MiniLM)

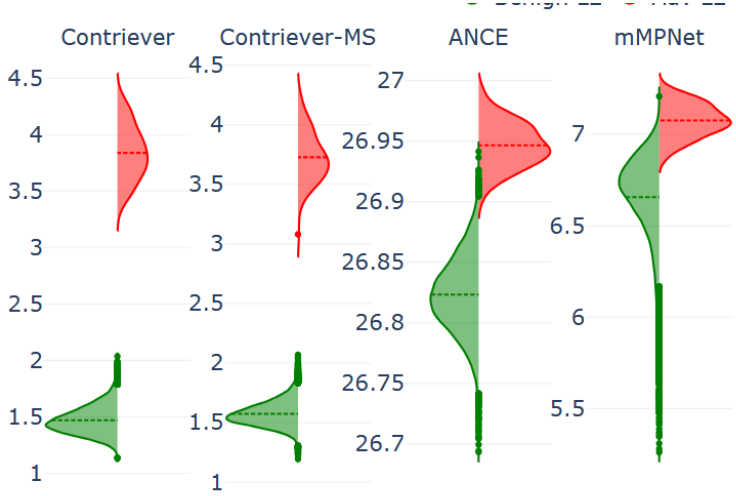
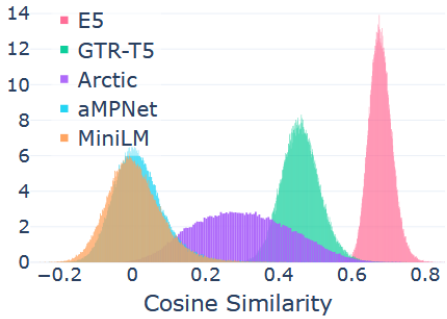
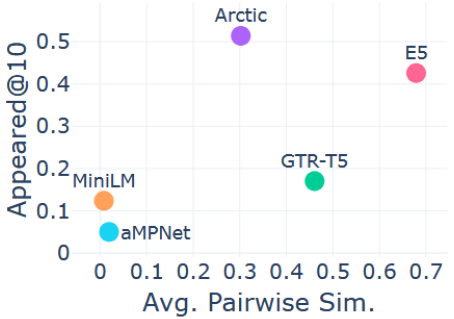


Figure 10: ℓ_2-norm distribution of benign and adversarial passages, crafted in §6.2.



(a) Similarities of random text pairs



(b) vs. Attack Success (`appeared@10`)

Figure 11: Assessing the *anisotropy* of different embedding spaces (i.e. non-zero expected cosine similarity of random text pairs; Fig. 11a) and its relation to the attack success rate (Fig. 11b; `GASLITE` in §6.2, for $|\mathcal{P}_{adv}|$=1).

Pros, Cons, Takeaways

- Pros
  - First “realistic” evaluation on data poisoning attacks for RAG
  - Empirical evaluation on various retrievers, scenarios, datasets
  - Well-written, makes sense!
- Cons
  - White-box attack
  - Knows-Nothing attack is actually query-dependent attack (attacker briefly know distribution)
- Takeaways
  - Dense embedding-based retrievers can be heavily poisoned with tiny SEO-style injections
  - A carefully designed white-box attack (GASLITE) reliably breaks many popular retrievers
  - Need for robustly-trained retrievers

Thank you

Background: How to make text move closer to embedding?

- Query stuffing: Repeat target query in adversarial passage
- Hotflip: find optimal text substitutions in discrete text space to maximize similarity to target query

Algorithm 1 HotFlip

Input: Initial text t; objective J; num_iters; top_k

```
x ← t
for iter from 1 to num_iters do
    Compute gradients w.r.t. token embeddings (all positions)
    for each position do
        Score every vocab token via gradient–embedding dot product
        Keep top_k tokens for this position
    end for
    Evaluate objective for all retained position–token candidates
    Pick best position–token; apply substitution to x
end for
```

Output: Optimized text x

Background: How to make text move closer to embedding?

- Corpus Poisoning: Hotflip variant, but in multi-query setting

Algorithm 1 Corpus Poisoning via Gradient-Based Attack

Input: Retrieval model R; queries Q; corpus P; budget B; top_k

```
Encode queries with R
Cluster query embeddings into B groups
Adv set ← empty
for each cluster do
    Init adversarial passage with random tokens
    repeat
        for each token position do
            Compute gradient of summed cluster similarity w.r.t. position
            Score all vocab tokens via gradient–embedding dot product
            Keep top_k tokens
            Select best token by exact summed similarity
            Update passage at position
        end for
    until convergence or step limit
    Add passage to Adv set
end for
P_poisoned ← P with Adv set inserted
```

Output: P_poisoned