

Evaluating the Effectiveness and Efficiency of Demonstration Retrievers in RAG for Coding Tasks

Pengfei He, Shaowei Wang, Shaiful Chowdhury University of
Manitoba Winnipeg, Canada hep2

Tse-Hsun Chen Concordia University Montreal, Canada
peterc@encs.concordia.ca

背景

RAG(Retrieval-Augmented Generation)

- LLMに外部知識を動的に与えながら，生成を行う手法

「LLMに必要なときだけ外部辞書を見せてから
答えさせる」仕組み

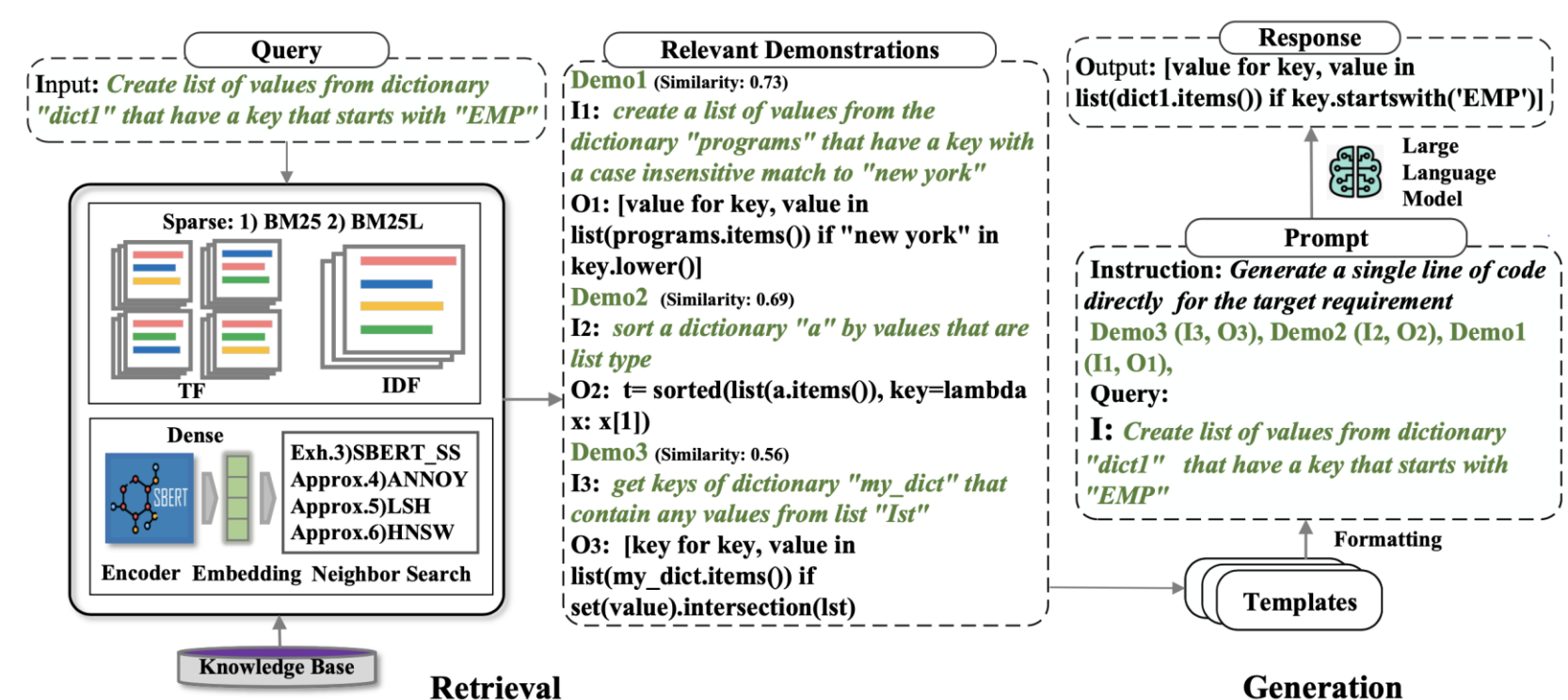
メリット

- ハルシネーションの低減
- モデルの学習データにない専門的な情報等も利用できる

構造

- Retrieval(検索) フェーズ
- Generation(生成) フェーズ

RAGの基本的な動作



- ユーザーの要求 → 関連例を検索 → デモを作成 → プロンプト化 → LLMでコード生成

背景

Retriever(検索器)

- 入力に対して、データベース内の文書から関連度の高いものを抽出する機構

主な検索手法

- スパースリトリバー：キーワード検索
 - tf-idf, BM25
- デンスリトリバー：ベクトル検索
 - ANN(近似最近傍探索)

ANN

- 完全に正しい最近傍を見つけるよりも、高速に解を求める手法

問題

RAGの運用

- 生成結果の精度と検索時間のバランスが重要で、互いにトレードオフの関係にある。

既存研究

- 生成精度を重視している研究ばかりであり、検索効率とのバランスを測っている研究がない。
- RAGを使用したANN検索(ベクトル検索)の研究は未だされていない。

目的

RAGを用いたコーディングタスクにおいて、異なる検索手法の精度と効率のトレードオフを明らかにする

RQ

- RQ1 : 有効性の評価
- RQ2 : 効率性の評価
- RQ3 : 有効性と効率性のトレードオフ
- RQ4 : 検索器のハイパーパラメータは効率性と有効性にどのような影響を与えるか
- RQ5 : ショット数の影響

目的

RQ1：有効性の評価

- コーディングタスクにおいて、明確な正解データが存在しない為、出力結果に近い検索結果(デモ)を正解データとし、評価.

RQ2：効率性の評価

- クエリに対してデモンストレーションを返すのに要する時間を計測(Time Per Query)

RQ3：有効性と効率性のトレードオフ

- BM25と比較した他のリトリバーとのスピードアップ率 および 生成品質の低下 を算出

目的

RQ4：検索器のハイパーパラメータは効率性と有効性にどのような影響を与えるか

- 近似リトリバーにおいて、ハイパーパラメータを変更することで近似の度合いが制御できる。

RQ5：ショット数の影響

- 例として与えるショットの数が大きく影響するのはわかっていたが、各リトリバーに最適なショット数を調査。

実験

使用LLM： **CodeLlama-Instruct-13B**

対象となる検索手法

- スパースリトリバー：BM25, BM25L
- デンスリトリバー：SBERT SS(非近似型),
ANNOY(木構造), LSH(ハッシュ), HNSW(グラフ)

データセット

- プログラム合成：CoNaLa データセット
- コミットメッセージ生成：NNgen データセット
- アサーション生成：Atlas データセット

実験

評価指標

- プログラム合成
 - Code Bleu , Syntax Match, Dataflow Match, N-gram Match rate
- コミットメッセージ生成
 - RougeL, Meteor, Rouge1, Rouge2
- アサーション生成
 - Exact Match, Plausible Match, Longest Common Subsequence, Edit Distance

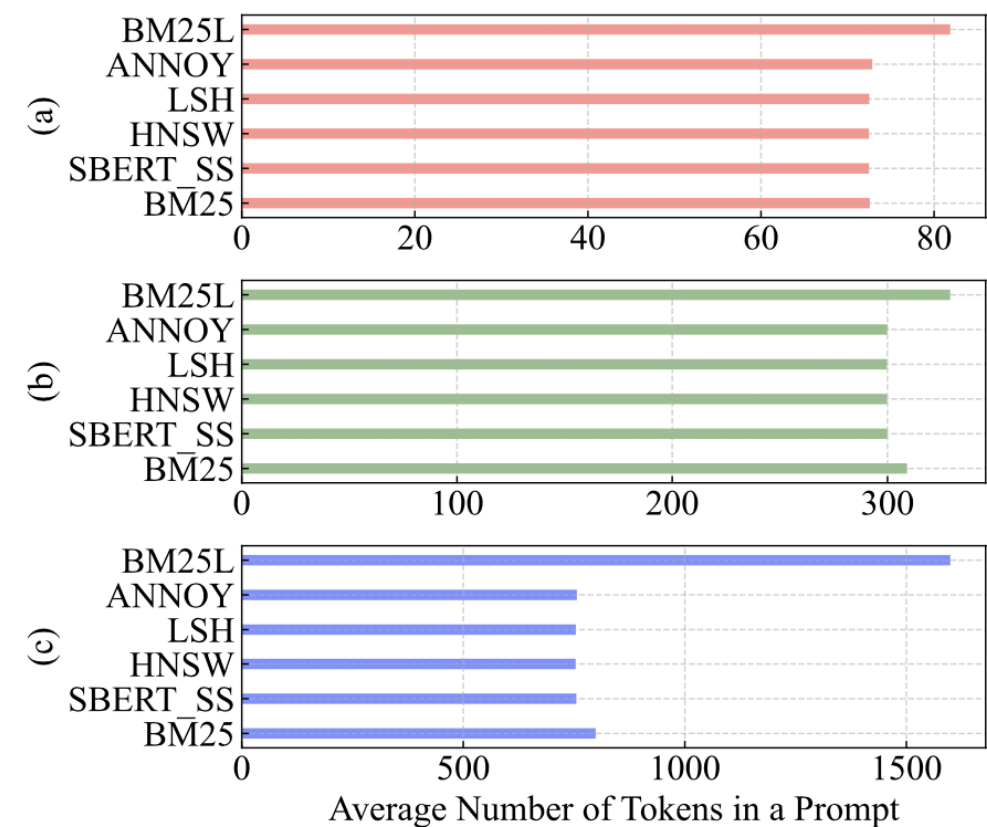
RQ1:有効性の評価

出力精度の正確さ

BM25 > SBERT SS (非近似型) $\geq$
ANNOY, LSH, HNSW (近似型) > BM25L

BM25Lが精度低い理由

- より長い文書を選びがち
 - 結果として、関連性よりも長さでデモを選んでしまい、有効性を失っている可能性がある。



(a) Program Synthesis (b) Commit Generation (c) Assertion Generation

RQ 2 : 効率性の評価

結果

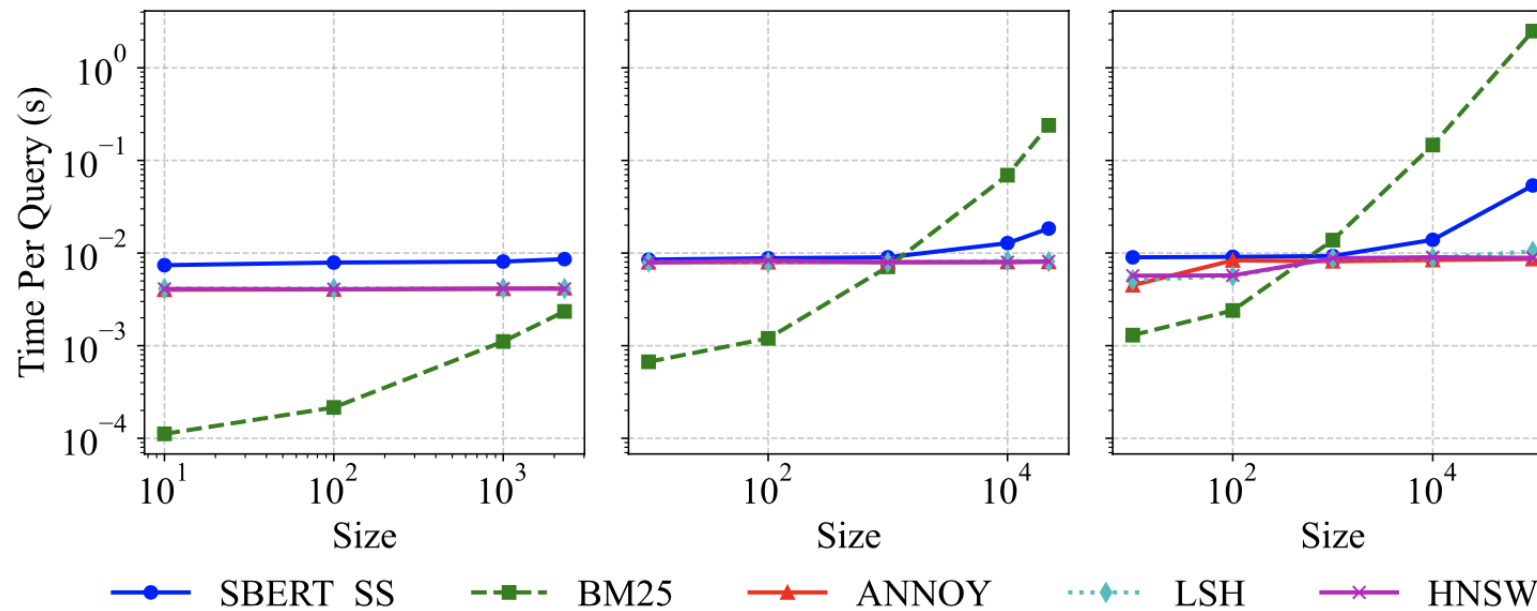
(a) Approach		Code Bleu	Syntax Match	Dataflow Match	Ngram Match	Time Per Query (ms)	Speedup	Reduced Code Bleu
Sparse	BM25L [14]	0.249	0.379	0.361	0.128	2.31	0.98x	5.75%↓
	BM25 [21]	0.264	0.391	0.373	0.146	2.26	1.00x	0.00%-
Exhaustive	SBERT [11]	0.261	0.387	0.380	0.138	5.07	0.45x	1.21%↓
Approximate	ANNOY [16]	0.259	0.380	0.385	0.136	5.29	0.43x	1.89%↓
	LSH [25]	0.252	0.385	0.376	0.124	5.08	0.44x	4.62%↓
	HNSW [18]	0.258	0.384	0.381	0.134	5.18	0.44x	2.27%↓
(b) Approach		RougeL	Meteor	Rouge1	Rouge2	Time Per Query (ms)	Speedup	Reduced RougeL
Sparse	BM25L	0.186	0.149	0.192	0.087	252.98	0.95x	35.34%↓
	BM25	0.288	0.244	0.294	0.169	239.53	1.00x	0.00%-
Exhaustive	SBERT_SS	0.286	0.238	0.290	0.166	16.67	14.37x	1.11%↓
Approximate	ANNOY	0.280	0.234	0.286	0.164	5.83	41.08x	2.74%↓
	LSH	0.285	0.242	0.291	0.168	5.82	41.17x	0.88%↓
	HNSW	0.283	0.237	0.289	0.166	5.44	44.05x	1.74%↓
(c) Approach		Exact Match	Longest Common Subsequence	Plausible Match	Edit Distance	Time Per Query (ms)	Speedup	Reduced Exact Match
Sparse	BM25L	0.414	73.3	0.492	15.9	206.19	1.15x	32.22%↓
	BM25	0.612	83.6	0.671	7.5	237.53	1.00x	0.00%-
Exhaustive	SBERT_SS	0.603	83.2	0.669	8.1	97.09	2.45x	1.32%↓
Approximate	ANNOY	0.576	81.8	0.649	8.9	6.34	37.45x	5.74%↓
	LSH	0.601	83.0	0.666	8.0	7.79	30.51x	1.78%↓
	HNSW	0.597	82.7	0.664	8.3	6.06	39.21x	2.35%↓

(a) プログラム合成 (b) コミット生成 (c) アサーション生成

- プログラム合成タスクを除いて，密ベクトル型リトリバーが優位
- タスクごとに速度が異なる。
 - 外部知識のサイズについて調査.

RQ 2 : 効率性の評価

スケーラビリティの調査

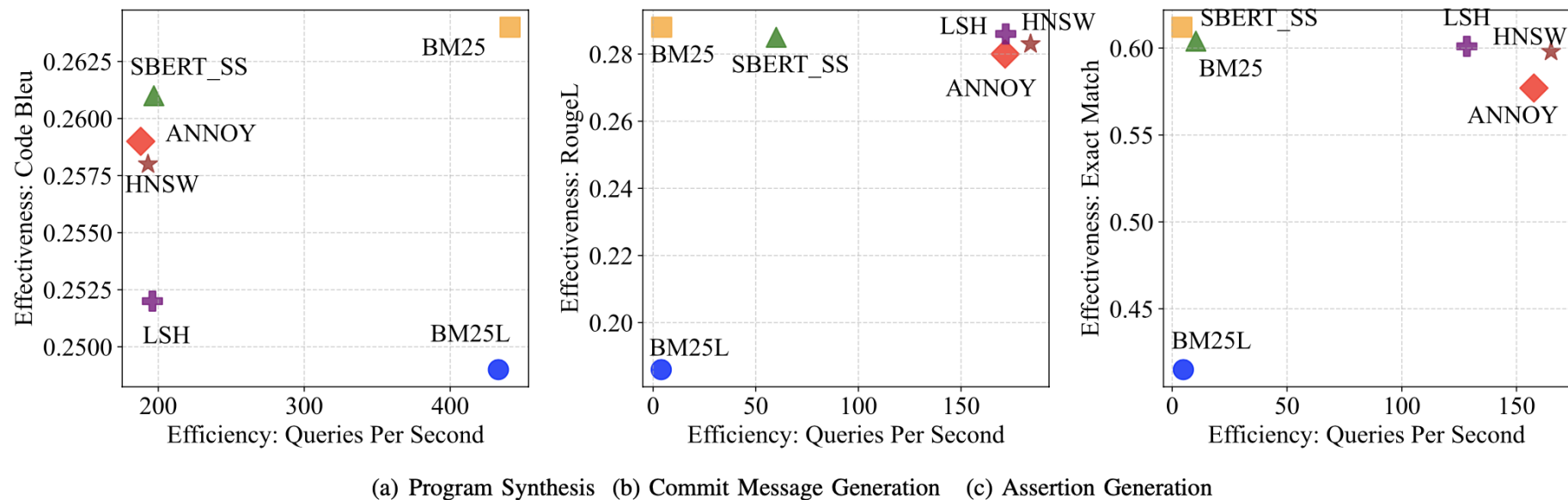


(a) Program Synthesis (b) Commit Generation (c) Assertion Generation

- 知識ベースが 10^3 未満の場合
スパース > 非近似型デンス > 近似型デンス
- 10^3 を超える知識ベース
近似的デンス > 非近似的デンス > スパース

RQ 3 : 効率性と有効性のトレードオフ

結果



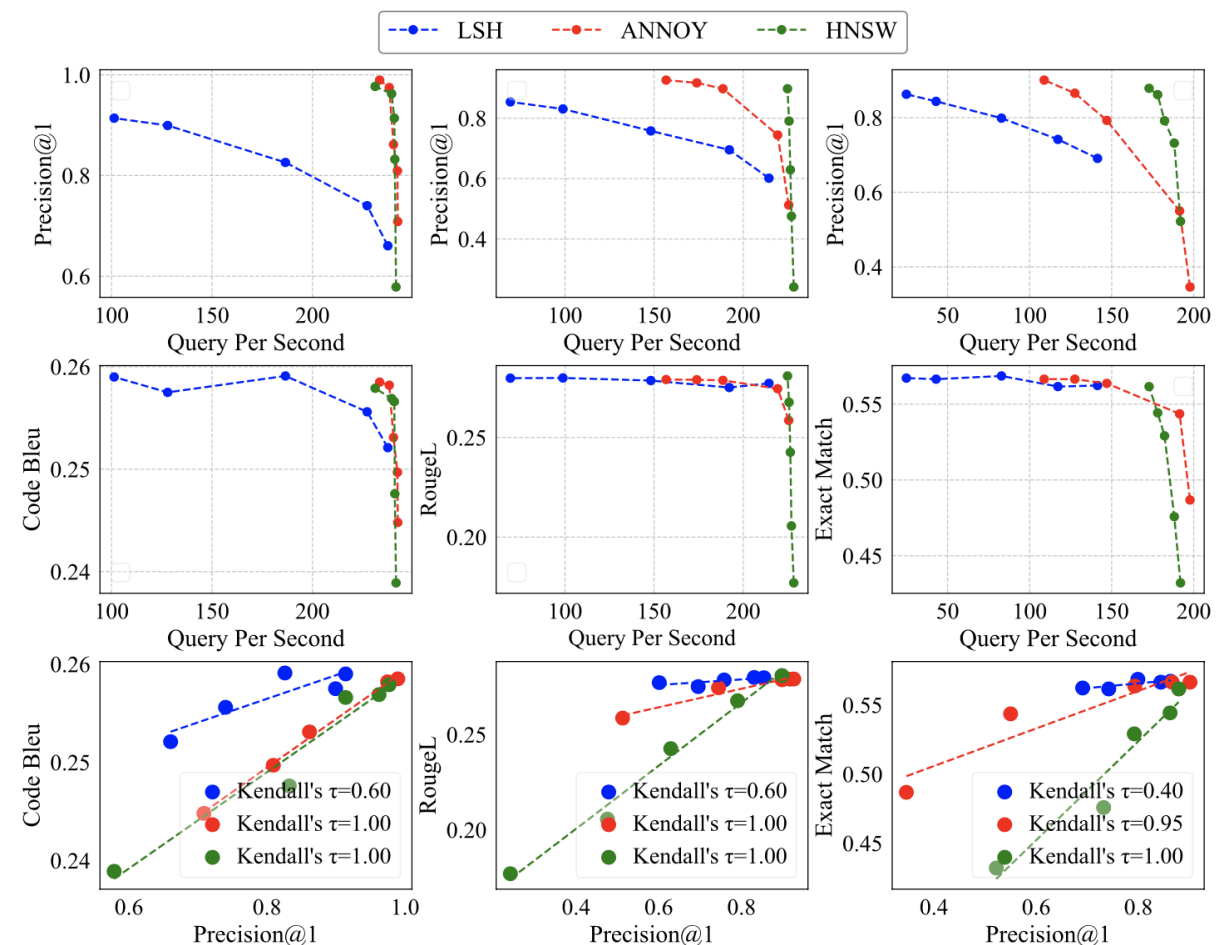
- BM25 : 最も有効であるが, 効率が悪い.
プログラム合成タスクでは両方優位.
- 大規模な知識ベースの場合, 近似リトリバーの方が優れている.

RQ4：ハイパーパラメータの影響

結果

- 近似をするほど速度は速くなるが，精度と効果を低下する。

- 検索精度(縦軸)と速度(横軸)
高速化すると，精度が下がる
- 生成精度(縦軸)と速度(横軸)
高速化すると，出力品質も下がる
- 生成精度(縦軸)と検索精度(横軸)
検索精度が高いほど，出力品質も下がる

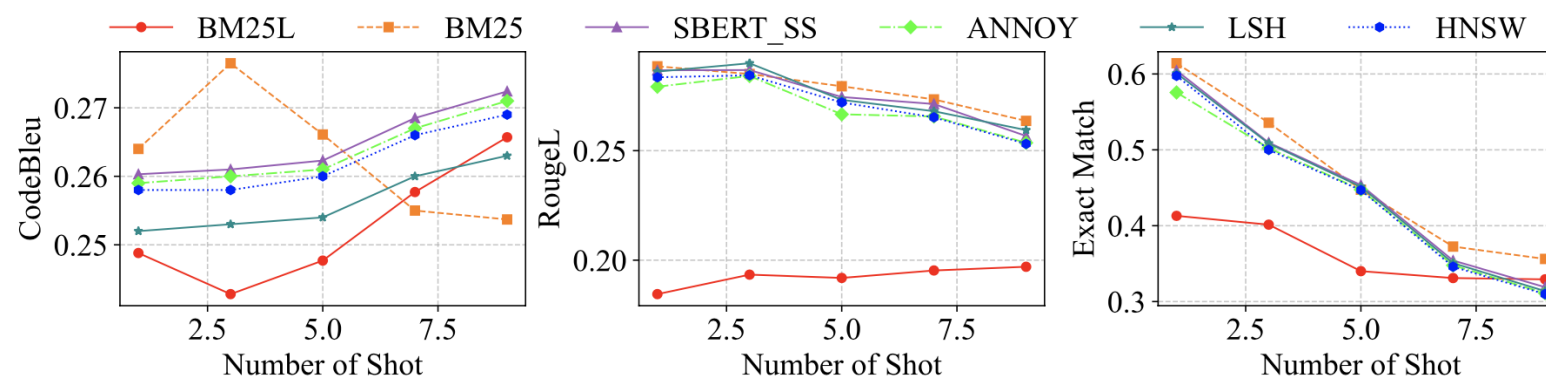


(a) Program Synthesis (b) Commit Generation (c) Assertion Generation

RQ5：ショット数の影響

結果

- デモンストレーション数が増えても、必ずしも有効性の向上に繋がるというわけではない。
- 理由
 - 多数のデモがノイズを導入してしまい、LLMが混乱してしまう
 - 特にデモの入力が似ているにもかかわらず、出力が大きく異なるときに顕著。



(a) Program Synthesis (b) Commit Generation (c) Assertion Generation

結論

- | スパース型リトリバーが最も有効性において、優れている。
- | 効率と有効性のトレードオフに関して、知識ベースのサイズに依存する。
- | デモンストレーションの数を増やすことは、RAGの有効性を高めるものではない。

