クロスドメイン検索型質問応答のための ニューラル疎ベクトル検索

西田光甫 [1,2] 吉永直樹 [3] 西田京介 [1]
[1] 日本電信電話株式会社 NTT 人間情報研究所
[2] 東京大学
[3] 東京大学 生産技術研究所
{kosuke.nishida, kyosuke.nishida}@ntt.com, ynaga @ iis.u-tokyo.ac.jp

概要

検索型質問応答はコーパスからの検索を用いて質問に回答するタスクである．質問応答で機密性の高いドメインを扱う場合，ターゲットドメインの文書を訓練時に使えない・ユーザ環境で運用する必要がある，の制約がある．そこで本研究では，訓練と推論のドメインが異なるクロスドメイン検索型質問応答タスクに，CPU で高速に動作する疎ベクトル検索で取り組んだ．提案手法は，ソースドメインで学習したモデルの埋め込み行列をターゲットドメインに低コストに適応させ，新たな学習可能パラメータの導入によって出力のスパース性を強めることで，2 つの制約に対処する．評価実験により，精度と計算効率双方における提案手法の有効性を確認した．

1 はじめに

検索型質問応答タスク [1] は，回答をモデルパラメータに基づき生成するのではなく，予め用意した文集合の中から回答を含む文を検索・提示することで応答するタスクである．検索型質問応答は，制御の難しいテキスト生成を介さないため，出力の制御可能性に優れる．

質問応答で機密性の高いドメインを扱う場合，ターゲットドメインの文書を訓練時に使えない・ユーザデバイスなどのローカル環境で運用する必要がある，といった実用上の制約がある．そこで本研究では，第一の制約から，訓練と推論のドメインが異なるクロスドメイン検索型質問応答に取り組んだ．さらに第二の制約から，inverted index を用いることで CPU でも高速に動作する疎ベクトル検索の手法に着目した．検索型質問応答の既存研究 [2, 3] では疎ベクトル検索のニューラル化が提案されており，ニューラルネットを要する演算を inverted index 作成時に全て済ませることで，高速な推論を実現する．一方，第一の制約が課題として残っている．

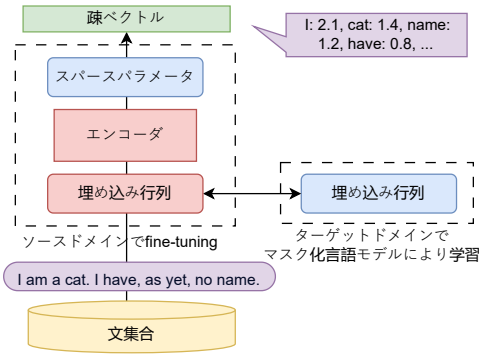

図 1 提案手法 SPARC の概要図．

そこで，本論文では 2 つの制約の解決を図るモデル SPARC（sparse neural retrieval model for cross-domain retrieval）を提案する．SPARC は図 1 に青枠で示す 2 つの特徴を持つ．第一に，モデルの埋め込み行列をターゲットドメインで訓練した行列に入れ替える．ターゲットドメインにおける埋め込み行列はマスク化言語モデル [4] によって得る．訓練のステップ数を制限することで，ターゲットドメインでの訓練を CPU 上で実施する．第二に，モデル出力のスパース性を強める学習可能なパラメータを導入することで，スパース制約下でのモデル性能を向上する．

MultiReQA [5, 2] を用いた評価にて，SPARC がソース・ターゲットドメイン双方で性能を向上し，密ベクトル検索より高速に動作することを確認した．

2 クロスドメイン検索型質問応答

クロスドメイン検索型質問応答を以下に定義する．本タスクはソース D_s とターゲットドメイン D_t を持つ．本タスクの目標は，あるドメイン $d \in D_t$

の文集合 S_d に基づいて，質問 q の回答を含む文 s を S_d から検索することである．

本研究は実サービスを念頭に以下のプライベートな質問応答システムを想定する．まず，サービス提供者がソースドメインの文集合を用いて GPU 上でモデルを訓練する．次に，各ユーザがターゲットドメインの文集合を CPU 上で inverted index に変換する前処理を行う．最後に，各ユーザが CPU 上で質問応答システムを動作させる．この処理は inverted index に基づき高速に動作する必要がある．

3 関連研究

3.1 質問応答における検索

検索タスクは質問応答タスクのサブタスクとして多く用いられる [6]．近年は，近似最近傍探索 [7, 8] に基づく密ベクトル検索による手法が主流である [9, 10, 11, 12]．しかし，近似最近傍探索は複数 CPU・GPU で高速に動く手法であり，疎ベクトル検索が単 CPU 環境では高速に動作する [3]．疎ベクトル検索は TF-IDF [13] や BM25 [14] が伝統的だが，近年は SPARTA [2] や SPLADE [15, 16, 17, 3] などのニューラルな手法が登場している．これらの手法は，従来法と異なり文脈情報を利用できる・単一のベクトル表現による密ベクトル検索と異なり単語レベルの情報を効率的に計算できる，の利点がある．

3.2 クロスドメイン質問応答

クロスドメイン質問応答の主流のアプローチではターゲットドメインの文書に関連する質問を生成することで訓練データを作成する [18, 19, 20]．しかし，訓練に十分な学習データを作成することは高コストかつ，ユーザ環境での実施は困難である．

4 既存手法：SPARTA

$\boldsymbol{H}$ を文エンコーダとして，$\boldsymbol{H}(s) \in \mathbb{R}^{l \times d}$ を文 s の隠れ状態とする．ここで，l は文長，d は埋め込み次元である．語彙 V 中のトークン v と文 s の関連度を関数 f で定める

$$f(v, s) = \log(\mathrm{ReLU}(\max_i \boldsymbol{H}(s)_i^\top \boldsymbol{e}_v) + 1). \quad (1)$$

ここで，$\boldsymbol{e}_v$ はトークン v の静的埋め込みベクトル（つまり，0 層目の表現）であり，$\boldsymbol{H}(s)_i \in \mathbb{R}^d$ は $\boldsymbol{H}(s)$ の i 番トークンに相当するベクトルである．文検索の際は，質問 q と文 s の関連度を関数 g で定める

$$g(q, s; f) = \sum_{v \in q} f(v, s). \quad (2)$$

訓練時は，質問とその正解文・負例文をサンプリングし，$g(q, s; f)$ をスコアとする Cross-Entropy 損失で正解文と in-batch negative を区別する学習を行う．

推論時は，$\{f(v, s)\}_{v \in V}$ を全ての文 $s \in S_d$ に関して計算する．f は質問に依存しないため，前処理として計算を完了できる．次に，各文 s について top-K スコアのトークンを inverted index に保存する．最後に，質問 q に対して $g(q, s; f)$ を inverted index に基づいて計算し，$g(q, s; f)$ の高い文を検索する．

5 提案手法：SPARC

SPARTA に埋め込み行列の置換とスパースパラメータを導入したモデル SPARC を提案する．

5.1 埋め込み行列の置換

モデルにターゲットドメインの知識を与える手法としては，モデルの fine-tuning の前にターゲットドメインにおける追加事前学習を行う手法が一般的である [21]．しかし，本研究の設定では fine-tuning 時にはターゲットドメインが未知であるため，この手法は採用できない．

そこで，図 1 に示すように，fine-tuning 後のモデルの埋め込み行列を，ターゲットドメインにおけるマスク化言語モデルで学習した行列と置換する手法を提案する．まず，モデルの fine-tuning は埋め込み行列を固定してモデルを更新する．次に，ターゲットドメインの知識を埋め込み行列に与えるため，元々の事前学習済みモデルを埋め込み行列以外を固定した状態でマスク化言語モデルにより更新する．最後に，fine-tuning 後のモデルの埋め込み行列をマスク化言語モデルによって得た行列に置換する．

本操作は過学習を抑える目的と CPU 上で計算を行う目的から，少量のステップ（実験では 500 ステップ）のみ学習する．埋め込み行列を置換してもモデルが破壊されない背景には，疎ベクトル検索とマスク化言語モデルの学習が，ともに隠れ状態 $\boldsymbol{H}(s)$ と正解トークンの埋め込み $\boldsymbol{e}_v$ の内積を大きくする過程であり，両タスクが類似していることがある．

5.2 スパースパラメータ

トークン・文の関連度 $f(v, s)$ (式 1) は $\log(\mathrm{ReLU}(x)+1)$ の関数から合成されており，この関数が負の x を

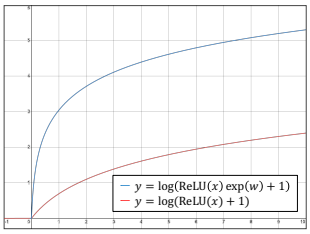

図 2 $y = \log(\mathrm{ReLU}(x) + 1)$ と $y = \log(\mathrm{ReLU}(x)\exp(w) + 1)$ のグラフ．$\exp(w) = 20$ とした．

零とすることで出力のスパース性を担保している．なお，対数関数を用いることは，この後に $f(v, s)$ に関する sum 演算をした値が Cross-Entropy における logit として扱われることから，自然な選択であると言える．しかし，図 2 に示す関数 $\log(\mathrm{ReLU}(x)+1)$ の勾配を考慮すると，正の小さな x のスパース化が促進されづらい・大きな x 間での関数値の差が小さく重要語の区別がされにくい，といった課題がある．

そこで，我々は学習可能なパラメータ w を導入してスケールの調整を行う

$$f(v, s) = \log(\mathrm{ReLU}(\max_i \boldsymbol{H}(s)_i^\top \boldsymbol{e}_v)\exp(w) + 1). \quad (3)$$

指数関数は値が正であること・学習中に通常のパラメータよりも大きい範囲を動くこと，の要請から用いた．実際に学習後の $\exp(w)$ の値は 20 を超えており，スパースパラメータの重要性を示している．図 2 に示すように，スパースパラメータの導入は関数 $\log(\mathrm{ReLU}(x) + 1)$ の課題を軽減する．

6 評価実験

6.1 実験設定

データセット 検索型質問応答データセットの MultiReQA [5, 2] を用いて評価実験を行った．MultiReQA は 12 個の機械読解データセットをまとめた MRQA [22] に質問・参照テキストの紐付けを削除するなどの前処理を施し，コーパスからの回答文検索タスクとして定式化することで作成された．具体的な前処理は付録に示す．前処理において検索型質問応答タスクとして成立しないと判断したデータセット（回答に複数文を要するデータなど）を除いた 5 つのデータセットを実験に用いた．データのドメインと統計値を表 1 に示す．

評価指標 MRR と Recall@1 について 3 回の実験の中央値を報告する．

実装 実験は事前学習済み言語モデル DistilBERT-base-uncased [28] で行った．なお，DistilBERT を用いた $\{f(s, v)\}_{v \in V}$ の計算は MacBookAir (M1, 16GB memory) 上で 1 文当たり 0.9 秒である．

6.2 比較手法

教師なし疎ベクトル検索として **BM25** [14] を，ニューラル疎ベクトル検索として SciPy [29] を用いて実装した **SPARTA** [2] を採用した．ニューラル密ベクトル検索として faiss [8] を用いて実装した Sentence-BERT (**S-BERT**)[1] [30] を採用した．S-BERT は state-of-the-art の手法ではないが，後続の研究は学習アルゴリズムを工夫する手法 [12] と計算量の増大を伴う手法 [31] が主流である．前者の手法は本研究の方向性と直交しており，SPARC にも適用できる．後者の手法は本研究の目的から外れる．

6.3 結果と議論

SPARC は検索性能を向上するか? 表 1 に主結果を示す．SPARC は SPARTA を BioASQ データ以外で上回った．埋め込み行列の置換を除いた場合も SPARC が性能を上回っているため，スパースパラメータの導入によって重要語のスコアが強まり，非重要語のスコアが抑制されていると言える．

埋め込み行列の置換による性能向上は TriviaQA・SearchQA・BioASQ データで見られる．SQuAD と Natural Questions データで向上がなかった背景には，SQuAD が訓練データであること，Natural Questions が SQuAD と同じ Wikipedia から作成されていることがあると考えられる．埋め込み行列の置換は訓練・ターゲットドメインの乖離を低減する効果がある．そのため，ユーザがユーザ自身の文書集合を扱う本研究の設定では効果的であると考える．

医療ドメインの BioASQ データでは，SPARTA が SPARC を上回った．医療ドメインはドメイン特有語を多く含む．SPARC は出力のスパース性を向上するため，ドメイン特有語を構成するサブワードのスコアを抑制してしまったと考えられる．スパース性は計算量の削減の効果があるが，計算量とドメイン特有語の多いドメインにおける検索精度にはトレードオフが存在すると言える．

SPARC は教師なし疎ベクトル検索手法の BM25 と密ベクトル検索手法の S-BERT を上回った．BM25 に比べると，SPARC は文の文脈情報を使う点に優れる．S-BERT に比べると，SPARC は質問・文を 1 つ

1) https://huggingface.co/sentence-transformers/distilbert-base-nli-mean-tokens

表 1 統計値と実験結果．全て SQuAD データセットで訓練し，括弧内の訓練データは未使用である．スコアは MRR/Recall@1 と表記した．Emb.Rep. は埋め込み行列の置換を示す．

	SQuAD [23]	Natural Questions [24]	TriviaQA [25]	SearchQA [26]	BioASQ [27]	Average
Domain	Wikipedia	HTML Wikipedia	Web snippets	Web snippets	Science articles	—
Queries	Crowdsourced	Search logs	Trivia	Jeopardy	Domain experts	—
$\|S_d\|$ for Train	95658	(448354)	(1893673)	(3163800)	—	—
$\|q\|$ for Train	86355	(104065)	(61687)	(117219)	—	—
$\|S_d\|$ for Eval.	10641	22117	238338	454835	14157	—
$\|q\|$ for Eval.	10477	4176	7784	16978	223	—
Unsupervised sparse model						
BM25	55.44/48.15	27.42/20.45	32.77/24.06	48.61/33.20	41.76/31.70	45.98/36.90
DistilBERT trained on SQuAD						
S-BERT	67.30/57.13	27.76/19.19	39.53/27.51	43.74/29.90	51.95/41.07	48.76/38.21
SPARTA	84.31/77.47	42.92/32.70	53.71/40.78	55.52/37.94	**75.84/64.73**	62.46/50.72
SPARC - Emb.Rep.	**84.97/78.12**	**45.09/34.85**	55.11/42.01	57.95/40.28	74.62/62.95	63.55/51.64
SPARC	84.96/78.02	44.97/34.62	**55.42/42.57**	**58.15/40.53**	75.10/63.39	**63.72/51.83**

表 2 文当たりの非零トークン数の中央値．

	SQuAD	Natural Questions	TriviaQA	SearchQA	BioASQ	Average	Std.
SPARTA	5316	3923	4167	5353	5805	4912.8	733.3
SPARC	1246	1120	1108	1189	1087	1150.0	58.9

表 3 検索性能とレイテンシ．top-1000 文の検索における計算時間を示す．SPARC と SPARTA の計算時間の違いは非零トークン数のみに依存するため記載しない．

	MRR	Recall@1	Latency[ms]
SPARC (K = 100)	59.48	69.68	0.69
SPARC (K = 500)	83.89	77.96	0.77
SPARC (K = 1000)	84.82	77.99	0.77
SPARC (K = 2000)	84.96	78.02	0.77
S-BERT (d = 768)	67.30	57.13	41.5

のベクトルで表すのではなくそれぞれのトークン間の相互作用を陽にモデリングする点に優れる．

スパースパラメータは出力をスパースにするか？ 疎ベクトル検索モデルのスパース性を調査するため，文当たりの非零トークン数を調べた．表 2 に結果を示す．式 3 に導入したスパースパラメータは単に $\boldsymbol{H}(s)_i$ と $\boldsymbol{e}_v$ の内積のスケールを調整しているのみであるが，結果として出力のスパース性が向上していることがわかる．また，非零トークン数の標準偏差が減少していることからも，SPARC のドメインに関する頑健性が示唆される．また，SPARC は SPARTA に比べて検索精度だけでなく計算効率の点からも優れていることがわかる．

SPARC は高速に検索できるか? 表 3 に SQuAD データセットにおける MacBook Air 上での検索精度とレイテンシを示す．SPARC は密ベクトル検索の S-BERT を計算精度とレイテンシの双方で上回っている．S-BERT は密ベクトル検索における state-of-the-art の手法ではないが，ColBERT [31] などの手法はさらに計算量が大きい．したがって，本研究の目的であるプライベート質問応答には，疎ベクトル検索は密ベクトル検索よりも適していると言える．なお，空間計算量は概ね非零トークン数に比例するため，表 2 の結果から SPARC は SPARTA より優れる．

7 おわりに

本論文ではクロスドメイン検索型質問応答に取り組んだ．本研究の目的は個人情報や社内情報を含む機密性の高いドメインを扱う検索型質問応答の実現であり，ターゲットドメインの文書を訓練時に使えない・ユーザデバイスなどの低資源環境で運用する必要がある，の課題がある．

本研究の独自性 上記の課題に対して，埋め込み行列の置換とスパースパラメータの導入を提案し，検索精度の向上・計算量の削減双方を確認した．

本研究の重要性 本研究は高効率でセキュアなプライベート質問応答システムの構築に貢献する．また，本研究はドメインレベルの適応を超えた質問応答のパーソナライゼーションに資する研究であり，質問応答システムの適用範囲の拡大に貢献する．

謝辞

本研究（第二著者）は東京大学生産技術研究所特別研究経費および JSPS 科研費 JP21H03494 の助成を受けています.

参考文献

[1] Amin Ahmad et al. ReQA: An evaluation for end-to-end answer retrieval models. In **MRQA**, pp. 137–146, 2019.

[2] Tiancheng Zhao et al. SPARTA: Efficient open-domain question answering via sparse transformer matching retrieval. In **NAACL**, pp. 565–575, 2021.

[3] Carlos Lassance and Stéphane Clinchant. An efficiency study for splade models. In **SIGIR**, SIGIR '22, p. 2220–2226, 2022.

[4] Jacob Devlin et al. BERT: Pre-training of deep bidirectional transformers for language understanding. In **NAACL**, pp. 4171–4186, 2019.

[5] Mandy Guo et al. MultiReQA: A cross-domain evaluation forRetrieval question answering models. In **Adapt-NLP**, pp. 94–104, 2021.

[6] Danqi Chen et al. Reading Wikipedia to answer open-domain questions. In **ACL**, pp. 1870–1879, 2017.

[7] Anshumali Shrivastava and Ping Li. Asymmetric lsh (alsh) for sublinear time maximum inner product search (mips). In **NIPS**, Vol. 27. Curran Associates, Inc., 2014.

[8] Jeff Johnson et al. Billion-scale similarity search with GPUs. **IEEE Transactions on Big Data**, Vol. 7, No. 3, pp. 535–547, 2021.

[9] Lee Xiong et al. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In **ICLR**, 2021.

[10] Vladimir Karpukhin et al. Dense passage retrieval for open-domain question answering. In **EMNLP**, pp. 6769–6781, 2020.

[11] Kenton Lee et al. Latent retrieval for weakly supervised open domain question answering. In **ACL**, pp. 6086–6096, 2019.

[12] Yingqi Qu et al. RocketQA: An optimized training approach to dense passage retrieval for open-domain question answering. In **NAACL**, pp. 5835–5847, 2021.

[13] Karen Spärck Jones. A statistical interpretation of term specificity and its application in retrieval. **Journal of Documentation**, Vol. 28, No. 1, pp. 11–21, 1972.

[14] Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: Bm25 and beyond. **Foundations and Trends® in Information Retrieval**, Vol. 3, No. 4, pp. 333–389, 2009.

[15] Thibault Formal et al. Splade: Sparse lexical and expansion model for first stage ranking. In **SIGIR**, p. 2288–2292, 2021.

[16] Thibault Formal et al. Splade v2: Sparse lexical and expansion model for information retrieval. **CoRR**, Vol. abs/2109.10086, , 2021.

[17] Thibault Formal et al. From distillation to hard negative sampling: Making sparse neural ir models more effective. In **SIGIR**, SIGIR '22, p. 2353–2359, 2022.

[18] David Golub et al. Two-stage synthesis networks for transfer learning in machine comprehension. In **EMNLP**, pp. 835–844, 2017.

[19] Siamak Shakeri et al. End-to-end synthetic data generation for domain adaptation of question answering systems. In **EMNLP**, pp. 5445–5460, 2020.

[20] Hongyin Luo et al. Cooperative self-training of machine reading comprehension. In **NAACL**, pp. 244–257, 2022.

[21] Suchin Gururangan et al. Don't stop pretraining: Adapt language models to domains and tasks. In **ACL**, pp. 8342–8360, 2020.

[22] Adam Fisch et al. MRQA 2019 shared task: Evaluating generalization in reading comprehension. In **MRQA**, pp. 1–13, 2019.

[23] Pranav Rajpurkar et al. SQuAD: 100,000+ questions for machine comprehension of text. In **EMNLP**, pp. 2383–2392, 2016.

[24] Tom Kwiatkowski et al. Natural questions: A benchmark for question answering research. **TACL**, Vol. 7, pp. 452–466, 2019.

[25] Mandar Joshi et al. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In **ACL**, pp. 1601–1611, 2017.

[26] Matthew Dunn et al. Searchqa: A new q&a dataset augmented with context from a search engine. **CoRR**, Vol. abs/1704.05179, , 2017.

[27] George Tsatsaronis et al. An overview of the bioasq large-scale biomedical semantic indexing and question answering competition. **BMC bioinformatics**, Vol. 16, No. 1, pp. 1–28, 2015.

[28] Victor Sanh et al. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. In **EMC**^{2}, 2019.

[29] Pauli Virtanen et al. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. **Nature Methods**, Vol. 17, pp. 261–272, 2020.

[30] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In **EMNLP-IJCNLP**, pp. 3982–3992. Association for Computational Linguistics, 2019.

[31] Omar Khattab and Matei Zaharia. ColBERT: Efficient and effective passage search via contextualized late interaction over bert. In **SIGIR**, p. 39–48, 2020.

[32] Zhilin Yang et al. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In **EMNLP**, pp. 2369–2380, 2018.

[33] Aniruddha Kembhavi et al. Are you smarter than a sixth grader? textbook question answering for multimodal machine comprehension. In **CVPR**, 2017.

[34] Omer Levy et al. Zero-shot relation extraction via reading comprehension. In **CoNLL 2017**, pp. 333–342, 2017.

[35] Adam Paszke et al. Pytorch: An imperative style, high-performance deep learning library. In **NeurIPS**, pp. 8024–8035. Curran Associates, Inc., 2019.

[36] Thomas Wolf et al. Transformers: State-of-the-art natural language processing. In **EMNLP System Demonstrations**, pp. 38–45, 2020.

[37] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In **ICLR**, 2015.

A　MultiReQA データセット

本節では MultiReQA データセットの前処理を述べる．機械読解データセットは参照テキスト・質問・回答のトリプルで構成される．まず参照テキストを文分割し，回答を含む文を正解文とする．各データセットのスプリットごとに全ての文をマージし，文集合とする．質問に代名詞が使われているなどの理由で，参照テキストと質問の紐づけを削除することで質問の意図が不明瞭になることがある．既存研究 [5] に従い，多くの質問の意図が不明瞭になる 4 個のデータセットを削除した．

さらに我々は，既存研究がモデル性能を過小評価していることを発見した．一部のデータセットは全く同じ複数の質問を含み，それらは異なる参照テキストに紐づいている[2]．これらはファクトイドな質問であるため，どの参照テキストの回答を選んでも本来は正解である．そこで，完全に一致する質問は 1 つの質問としてマージする処理を全てのデータセットで実施した．さらに，以下のデータセットを削除した．HotpotQA [32] はマルチホップな推論を含み単文での回答が不可能なため，削除した．TextBookQA [33] は回答に図表の視覚的な理解を要するため，削除した．Relation Extraction [34] は合成データセットであり性能も 95%を超えているため，削除した．結果として，我々が前処理した MultiReQA データセットは 5 個のデータセットから構成される．

B　実験設定の詳細

既存研究に従い，文をエンコーダに入力する際は，単に文を入力するのではなく，文を含む元々の文章（段落など）を周辺文脈として入力する [2]．すなわち，文 s はエンコーダに ‘[CLS] 文 s [SEP] 元々の文章 [SEP]’ として入力する．なお，[CLS] と [SEP] は特殊トークンである．入力は最初の 256 トークンのみを用いる．

Cross-Entropy 損失で用いる質問に対する負例文は以下の方法でサンプリングする．まず，正解文を含む文章の正解文以外をランダムに 1 つ選ぶ．また，BM25 [14] を用いた検索を行い，top-100 の文を抽出する．top-100 文から質問の回答を含む文を取り除き，残った文からランダムに 1 つ選ぶ．質問のバッチサイズを B としたとき，2 つの負例文を用いるため，in-batch negative は $3(B-1)$ である．

実装には PyTorch (ver. 1.12.1) [35][3] と transformers (ver. 4.21.1) [36][4] を用いた．最適化手法を Adam [37]，学習率を 1e-5，エポック数を 20，warmup step 数を 500 とした．スパースパラメータは通常のパラメータよりも大きな値になることを期待しているため，学習率を 1e-3 とした．実験には 4 枚の NVIDIA Quadro RTX 8000 (48GB) を用い，バッチサイズは 32（負例文を考慮すると 96）とした．文当たりの非零トークン数は $K = 2000$ でフィルタリングした．埋め込み行列の置換の際のマスク化言語モデルは学習率 5e-5, バッチサイズ 32 とした．

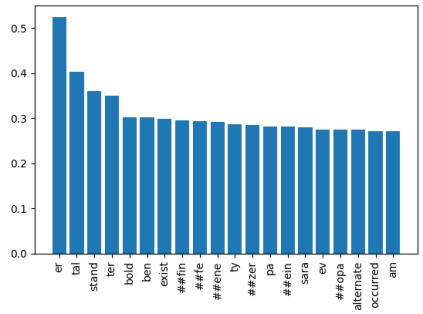

図 3　あるトークンが SPARTA では被零であったが SPARC では零になった文の数．縦軸は割合であり，文集合のサイズ $|S_d|$ で割った．top-20 のトークンのみを示す．

C　定性分析

スパースパラメータによってどのトークンのスコアが抑制されたか？　本節ではスパースパラメータによって抑制されたトークンの傾向を分析する．図 3 に，あるトークンが SPARTA では被零であったが SPARC では零になった割合を示す．

まず，top-20 トークンのほとんどがサブワードである．SPARC がある単語を構成する全てのサブワードに正のスコアを付けるのではなく，一部のサブワードのみにスコアを付けたと考えられる．これは，スパース制約下で効果的なスコア割当であると言える．

次に，存在を示す単語が多く含まれていることが観察できる（“stand”，“exist”，“occured”，“am”）．存在を示す語は文の意味に大きな影響を与えないことが多いと SPARC が学習し，スコアを割り当てないようになったと考えられる．

2)　例えば，「質問：ペルオキシレドキシン 2（PRDX2）はどのような酵素ですか？」「回答：抗酸化物質」など．BioASQ データセットから和訳して記載．

3)　https://pytorch.org/

4)　https://github.com/huggingface/transformers