高速・高精度検索のための勾配推定手法を用いた擬似スパースベクトルの学習法

長谷川拓　　西田京介　　加来宗一郎　　吉田仙
日本電信電話株式会社　NTT メディアインテリジェンス研究所
taku.hasegawa.ps@hco.ntt.co.jp

1　はじめに

深層学習技術の発展により，ニューラルネットワーク（NN）を用いた検索モデルの有効性について数多く報告されている [1]．近年では，BERT [2] をはじめとする事前学習モデルの利用により従来技術を大きく上回る精度で検索が可能となってきた [3]．

しかし，大規模検索においては，[3] の様にクエリと文書を併せて NN に入力する方式では，検索対象の文書数分の計算が検索のたびに必要となるため，検索速度の観点から実用的でない．そこで，クエリと文書を独立に NN で密ベクトルにエンコードして，最大内積探索（MIPS）により検索する方式が近年では盛んに研究されている [4]．MIPS は高速な検索を可能にするが，MIPS のインデックスを構築するための計算時間が膨大となる課題が残る．

そこで我々は，クエリと文書を独立に**スパースベクトル**にエンコードし，転置インデックスを利用可能にすることでインデックスの構築および検索の両方を高速に行うことを目指す．本研究では，長谷川らにより提案された α-SVS [5] の拡張に取り組む．α-SVS ではベクトルの各次元に対し，ミニバッチにおける絶対値の大きさの上位 100α%よりも値が小さな要素を 0 として得た擬似的なスパースベクトルを用いて損失を計算するが，離散的な forward 計算（top-α）の勾配が不安定となる問題があった．本研究では，離散 forward 関数の偏微分の線形補間により擬似的な勾配を推定する手法を導入することでこの問題を解決する．大規模コーパスである MS MARCO を用いた数値実験により，従来の α-SVS に比べ性能が向上したことを示す．

2　定義

本研究で取り組む情報検索と関連する用語について定義する．

定義 1 (擬似スパースベクトル). k 番目のベクトルを $v^k = [v^k_1, \ldots, v^k_D]^\top \in \mathbb{R}^D$ とする．v^k の擬似スパースベクトルの i 次元目を以下で定義する．

$$f(v^k_i|\theta_i) = \begin{cases} v^k_i & v^k_i > \theta_i, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

ここで θ_i は i 次元目の閾値である．

定義 2 (擬似スパースベクトルを用いた転置インデックス). 擬似スパースベクトルを用いた転置辞書を表現ベクトルの各次元を潜在単語とみなし，次元番号の集合 $\{1, 2, ..., D\}$ から $X^p \times \mathbb{R}_{>0}$ の部分集合の族への写像として定義する．ここで，インデックス i の像は文書とその文書とクエリの関連度スコアの部分集合 $\{(x^{p_k}, v^k_i) \mid k = 1, \ldots, K, v^k_i \neq 0\}$ である．

3　α-SVS

高次元のスパースベクトルを作ることにより，特徴ベクトルを擬似的に潜在単語とみなして転置インデックスを作成する手法 [6] がこれまで提案されてきたが，単純に L1 ノルムを最小化する方法では，ベクトルが部分空間に偏る問題があった．

長谷川らはこの問題を解決するために top-α 学習とこれを導入した検索モデル α-SVS を提案している [5]．図 1 に α-SVS のアーキテクチャを示す．

3.1　文脈埋め込み層

文脈埋め込み層の出力は $H = \mathrm{BERT}(X)$ として得られる．ここで，入力系列 X は，

$$X = ([\mathrm{CLS}], x^\cdot_1, x^\cdot_2, \ldots, [\mathrm{SEP}]) \tag{2}$$

とし，$\{x^\cdot\}$ はクエリ x^q あるいは文書 x^{p_k} のトークナイズされたサブワード系列である．BERT はクエリと文書に対して共有される．

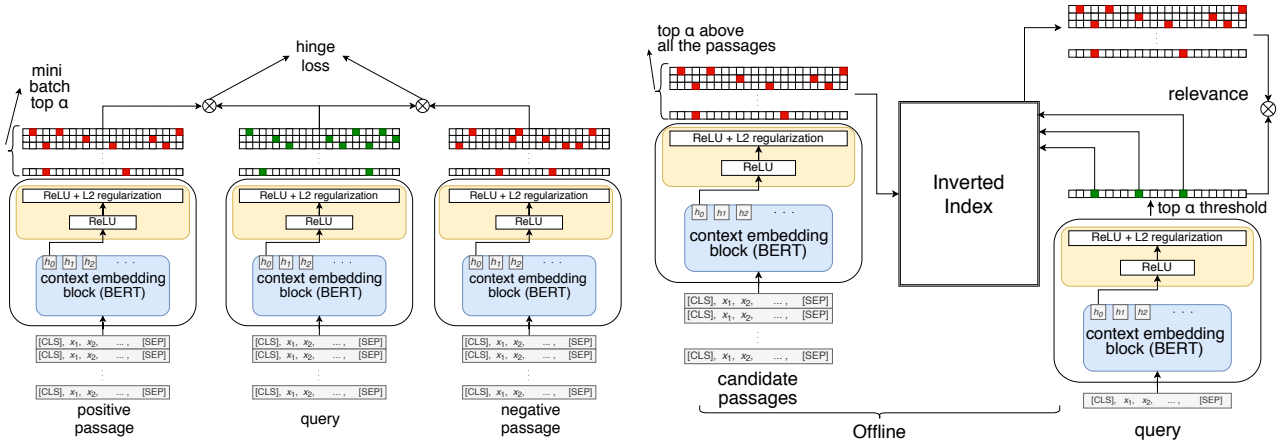

図 1　α-SVS のアーキテクチャ. 左図：訓練時のモデル．α-擬似スパースベクトルはクエリ，正例，負例のペアワイズで学習される．右図：検索時のモデル．α-擬似スパースベクトルで作成した転置辞書を用いてクエリと文書の内積で関連度スコアが算出される．

3.2　ベクトル出力層

ベクトル出力層は BERT の出力 H の [CLS] トークンに対応する h_0 を入力とし，全結合層 2 層と ReLU で高次元空間に写像した $v \in \mathbb{R}^D$ が出力となる．さらに，出力ベクトルの L2 ノルムが 1 となるように正規化を行い，超球面上で最適化を行う．

$$v' = \mathrm{ReLU}(W_2\mathrm{ReLU}(W_1 h_0 + b_1) + b_2), \quad v = \frac{v'}{||v'||_2}. \tag{3}$$

$W_2 \in \mathbb{R}^{D \times D_{hid}}$, $W_1 \in \mathbb{R}^{D_{hid} \times 768}$, $b_2 \in \mathbb{R}^D$ および $b_1 \in \mathbb{R}^{D_{hid}}$ はクエリ，文書のネットワークで共有される．

3.3　α-擬似スパースベクトル

α-SVS は得られたベクトル集合に対し，各次元の値の絶対値の大きさ上位 100 α% で閾値 θ_i（式 1）を決め，値が小さな要素を 0 とみなし α-擬似スパースベクトルを得る．数式による定義は付録 4 に記載した．この α-擬似スパースベクトルを用いて損失関数を計算して学習する方法を **top-α 学習**と呼ぶ．

また，検索時と学習時の上位 100α% の閾値はそれぞれ異なるベクトル集合から計算され，クエリと文書でもそれぞれ独立で閾値を計算する．訓練時の α は文書，クエリそれぞれ α_p，α_q と表し，検索時はそれぞれ α'_p，α'_q と表すこととする．

訓練時の α-擬似スパースベクトル　各ミニバッチでの与えられたクエリと文書のベクトル集合を V_i^q と V_i^p とする．α-擬似スパースベクトルは α_p，α_q を用いて以下のように計算される．

$$f_\mathrm{q}(v_i^\mathrm{q}) = \begin{cases} v_i^\mathrm{q} & v_i^\mathrm{q} > top(V_i^\mathrm{q}|\alpha_\mathrm{q}), \\ 0 & \text{otherwise} \end{cases}, \tag{4}$$

$$f_\mathrm{p}(v_i^\mathrm{p}) = \begin{cases} v_i^\mathrm{p} & v_i^\mathrm{p} > top(V_i^\mathrm{p}|\alpha_\mathrm{p}), \\ 0 & \text{otherwise} \end{cases}. \tag{5}$$

ここで，$\alpha_\mathrm{p} = \alpha_\mathrm{q} = 1$ の場合，密ベクトルによる学習と等価となる．

検索時の α-擬似スパースベクトル　検索時においては検索精度を保証できるような上位 α'_q を計算するのに十分なクエリ数が常に得られるとは限らない．ゆえに，α-SVS では，クエリについては訓練データのクエリ集合を用いて $top(V_i^\mathrm{q}|\alpha'_\mathrm{q})$ を計算し，文書については検索対象となる全ての文書集合から $top(V_i^\mathrm{p}|\alpha'_{\mathrm{p}'})$ を計算する．

3.4　損失関数

α-SVS ではペアワイズランキング学習を行う．訓練データとして $\mathcal{D} = \{(x_i^\mathrm{q}, x_i^{\mathrm{p}_1}, x_i^{\mathrm{p}_2}) \mid i = 1, \ldots, N\}$ を与える．ここで $x_i^{\mathrm{p}_1}$ はクエリの正例文書，$x_i^{\mathrm{p}_2}$ は負例文書である．i 番目の訓練サンプルに対するヒンジロス L_i は α-擬似スパースベクトルで計算した内積を用いて以下で定義される．

$$L_i = \max\{0, \epsilon - (f_\mathrm{q}(v_i^\mathrm{q}) \cdot f_\mathrm{p}(v_i^{\mathrm{p}_1}) - f_\mathrm{q}(v_i^\mathrm{q}) \cdot f_\mathrm{p}(v_i^{\mathrm{p}_2}))\} \tag{6}$$

ここで ϵ はヒンジロスのマージンを決定するハイパーパラメータである．

3.5　単一モデルでの 2 段階検索

α-擬似スパースベクトルは大規模検索で有用であるが，文書数が膨大な場合においては非常に小さい α を設定する必要がある．この場合 v で得られる関

連度スコアに対しての近似精度は落ちてしまうため，α-SVS では検索時に同じモデルを用いて２段階に検索を行う．１段階目は α-擬似スパースベクトルの内積で計算した関連度が上位 k 件の文書を選ぶ．２段回目では α-擬似スパースベクトル を作成する前の元のベクトル v を用いて内積で関連度スコアを計算し，最終的な関連文書のランキングを得る．従来では２段階検索と呼ばれる場合２つの検索モデルを用いて行われるが，α-SVS では単一のモデルで２段階検索を行うため，２段回目の検索時では クエリや文書をモデルに入力する必要がなく高速に計算可能である．

4 提案手法

前節で述べた α-SVS の学習をより精度良くするために，本論文では擬似勾配推定手法を導入する．擬似勾配推定手法はモデルの軽量化のためにモデルの低ビット化の際によく用いられる手法であり [7]，離散的な forward 計算が含まれるネットワークの学習を効果的に行うために用いられる．

4.1 top-α 学習中における勾配

top-α 学習の forward 計算のうち，式 1 で定義されている関数 f は偏微分を行うと

$$\frac{\partial f}{\partial v_i^k}(v_i^k|\theta_i) = \begin{cases} 1 & v_i^k > \theta_i, \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

となる．ただし $\theta_i = top(V_i|\alpha)$ である．

top-α 学習では，ミニバッチ内の他の要素次第で同じ文書ベクトルであっても top-α に含まれる場合と含まれない場合が存在しうる．式 7 で誤差逆伝搬法を用いて学習を行った場合、top-α に含まれなかった次元は誤差が伝搬されず V_i の選ばれ方に大きく影響を受ける．モデルの初期値にも依存するため，この不確定性は学習過程に影響を与えうる．

4.2 top-α 学習中の勾配推定

本論文では偏微分した関数を線形補完することにより，ミニバッチで選ばれる V_i の影響を軽減する手法を提案する．式 7 で示した top-α 学習での backward で用いられる関数を，以下の式で置き換えることにより勾配推定手法を導入する．

$$\frac{\partial g}{\partial v_i^k}(v_i^k|\theta_i,\theta_i') = \begin{cases} 1 & v_i^k > \theta_i, \\ \frac{1}{\theta_i-\theta_i'}v_i^k - \frac{\theta_i'}{\theta_i-\theta_i'} & \theta_i' < v_i^k \le \theta_i, \\ 0 & v_i^k \le \theta_i'. \end{cases} \tag{8}$$

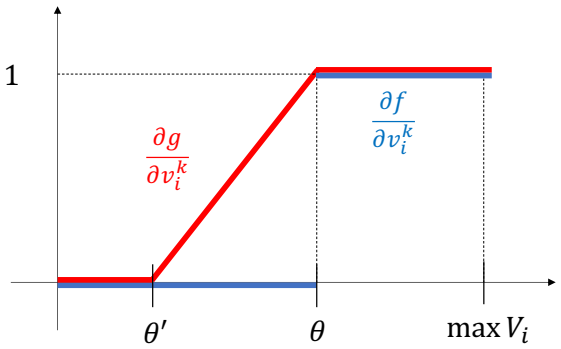

図 2 勾配推定の backward 関数

ここで，θ_i' は $\theta_i' \le \theta_i$ を満たす．図 2 に従来手法および提案手法で用いられる関数を示す．

ここで注意すべきは $\theta_i = top(V_i|\alpha)$ であり，V_i が与えられてから決定されるため，毎回異なる値をとる．θ_i' も同様に決定される必要があるため，本研究では下記の２通りの方法で θ_i' を決定した．

1. **V_i の最大値を用いた閾値.**

$$\theta_i' = 2top(V_i|\alpha) - \max V_i \tag{9}$$

2. **top-$k\alpha$ を用いた閾値.**

$$\theta_i' = top(V_i|k\alpha) \tag{10}$$

以後，本論文では 1 を用いたモデルを α−SVS$_{\max}$，2 を用いたモデルを α−SVS$_{k\alpha}$ とする．

5 評価実験

5.1 実験設定

MS MARCO 2.1 [9] の **Passage and Document Retrieval** タスクにより実験を行った．データの詳細および学習設定については付録 A.2 に記載した．

比較対象として，クエリから答えとなる文書を事前学習モデル T5 [10] を用いて予測し転置辞書によりキーワード検索を行う手法である docTTTTTquery [8] およびパッセージおよび文書レベルの単語重みをニューラルネットを用いて計算し，これらを利用しキーワード検索を行う手法 DeepCT [11, 12] を用いた．

5.2 勾配推定手法の効果に関する評価

α-SVS および勾配推定手法を導入した α-SVS のランキング精度および検索速度について評価し比較した．

表 1 に実験結果を示す．MSMARCO のリーダーボードでの評価指標として採用されている MRR @

表 1　MSMARCO パッセージ検索タスクにおける検索精度の結果．比較手法の結果は文献 [8] から引用．

	MRR@10 Dev	R@100 Dev	R@1000 Dev	Latency (ms/query)
BM25 (Anserini)	0.184	0.658	0.853	55
doc2query, top-k, 10 samples	0.218	-	0.891	61
DeepCT	0.243	-	0.913	55
docTTTTTquery, top-k, 40 samples	0.277	0.819	0.947	64
α-SVS	0.263	0.691	0.785	350
α−SVS$_{\max}$	**0.280**	0.708	0.765	-
α−SVS$_{2\alpha}$	0.270	0.677	0.728	-
BM25 + α-SVS	-	0.791	0.925	-
docTTTTTquery + α-SVS	-	**0.842**	**0.951**	-

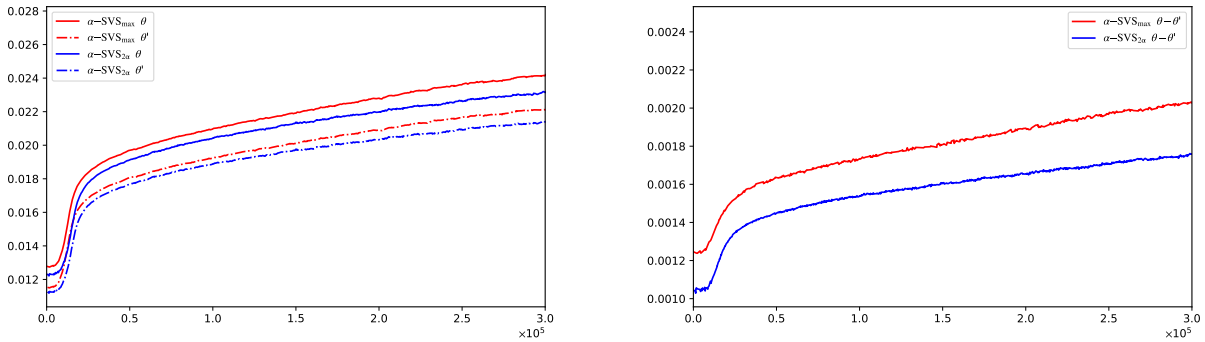

図 3　2 種類の勾配推定手法の 2 つの閾値とその差の推移．

10 では提案手法の α−SVS$_{\max}$ が最も良い結果となった．ただし docTTTTTquery や DeepCT と比べ Recall は低く，R@1000 については BM25 にも劣っている．このことから α-SVS は関連文書を大量の文書から上位にランキングする能力には長けているが，一部検索漏れを起こしうるということがわかった．特にこの問題は勾配推定手法を用いたとしても大きくは改善されず，さらに勾配推定手法を導入した α-SVS は従来の手法にくらべて僅かながら recall が下がっている箇所が見られる．よって勾配推定手法を導入した場合，上位のランキング精度は向上するが，Recall への貢献はみられず，劣化する可能性もあることがわかった．この問題への取り組みは今後の課題である．

一方で，docTTTTTquery で得られた結果と α-SVS で得られた結果を半分ずつ合わせた結果である docTTTTTquery + α-SVS の R@100 および R@1000 の結果は docTTTTTquery と比較しても最も良い結果であった．この結果から，α-SVS は他の手法では検索漏れを起こしうるパッセージを検索することが可能であることがわかり，α-SVS の有用性が示された．

5.3　2 つの勾配推定手法の比較

2 つの勾配推定の手法を比較するために学習中の θ および θ' の推移を調べた．図 3 に結果を示す．この結果から α−SVS$_{\max}$ は α−SVS$_{2\alpha}$ に比べ各次元の値の上位 100α% の閾値が高くなりやすい傾向があり，学習が進むにつれ線形補完される区間が大きくなりやすい傾向があることがわかった．これはあるベクトルは特定の次元に大きい値を持ちやすいことを示しており，α-SVS の出力は L2 ノルムが 1 であることから，α−SVS$_{\max}$ の方がよりスパース性の高いベクトルが得られやすい傾向にあることがわかった．

6　おわりに

本研究では大規模検索モデル α-SVS の学習をより促進するための勾配推定手法を提案し，MS MARCO 2.1 の passage ranking タスクにおいて従来手法を上回る性能を達成した．

従来の α-SVS ではランダムに選ばれるバッチ内での値の上位で閾値決まるため，学習の安定性に欠けるという問題があった．本研究では，勾配推定手法を導入することによりバッチによる閾値の揺らぎを考慮してより効率の良い学習を実現した．

参考文献

[1] Bhaskar Mitra and Nick Craswell. An introduction to neural information retrieval. *Foundations and Trends in Information Retrieval*, Vol. 13, No. 1, pp. 1–126, 2018.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL*, pp. 4171–4186, 2019.

[3] Rodrigo Nogueira and Kyunghyun Cho. Passage re-ranking with BERT. 2019. arXiv:1901.04085.

[4] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick S. H. Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *EMNLP*, pp. 6769–6781, 2020.

[5] 長谷川拓, 西田京介, 加来宗一郎, 富田準二. 高速な情報検索に向けた文脈考慮型スパース文書ベクトルの獲得. 人工知能学会全国大会論文集, Vol. 2020, pp. 4Rin145–4Rin145, 2020.

[6] Hamed Zamani, Mostafa Dehghani, W. Bruce Croft, Erik G. Learned-Miller, and Jaap Kamps. From neural re-ranking to neural ranking: Learning a sparse representation for inverted indexing. In *CIKM*, pp. 497–506, 2018.

[7] Zechun Liu, Baoyuan Wu, Wenhan Luo, Xin Yang, Wei Liu, and Kwang-Ting Cheng. Bi-real net: Enhancing the performance of 1-bit cnns with improved representational capability and advanced training algorithm. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[8] Rodrigo Nogueira and Jimmy Lin. From doc2query to doctttttquery. https://cs.uwaterloo.ca/~jimmylin/publications/Nogueira_Lin_2019_docTTTTTquery-v2.pdf, 2019.

[9] Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. MS MARCO: A human generated machine reading comprehension dataset. 2018. arXiv:1611.09268v3.

[10] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *CoRR*, Vol. abs/1910.10683, , 2019.

[11] Zhuyun Dai and Jamie Callan. Context-aware sentence/passage term importance estimation for first stage retrieval. *CoRR*, Vol. abs/1910.10687, , 2019.

[12] Zhuyun Dai and Jamie Callan. An evaluation of weakly-supervised deepct in the trec 2019 deep learning track. In *TREC*, 2019.

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

A 付録

A.1 定義

定義 3 (2段階検索タスク). クエリ x^{q} と K 個の文書 $X^{\mathrm{p}} = \{x^{\mathrm{p}_k} \mid k = 1, \ldots, K\}$ が入力として与えられ、初めにモデルは X^{p} からクエリと関連する T_1 個の文書の順序集合を返す．次にモデルは先の T_1 件の文書をリランキングし，T_2 個のクエリに関連する文書の順序集合を返す．ただし，順序集合とはクエリと文書の関連度によって順序づけられた集合である．

定義 4 (α-擬似スパースベクトル). ベクトル集合の i 番目の要素の集合 V_i を $V_i = \{v_i^k \mid k = 1, \ldots, \}$ とする．α-擬似スパースベクトル v^k の i 次元目は

$$f'(v_i^k, V_i|\alpha) = \begin{cases} v_i^k & v_i^k > top(V_i|\alpha) \\ 0 & (\text{otherwise}) \end{cases} \tag{11}$$

ここで，$top(\cdot|\alpha)$ は与えられた集合の上位 100 α% の閾値を返す関数である．

A.2 実験設定の詳細

データセット． MS MARCO 2.1 [9] の **Passage and Document Retrieval** タスクにより実験を行った．本タスクでは ReRanking と Full Ranking が Passage 単位，Document 単位それぞれで存在しており，本実験では Passage Ranking を用いて評価を行った．ReRanking タスクではあらかじめ BM25 を用いて絞り込まれた上位 1000 件の passage が与えられているが，Full Ranking では約 880 万件の検索対象文書から検索を行うことが求められる．ランキング指標としては Mean Reciprocal Rank (MRR) Craswell09a が用いられている．

学習用設定． Train Triples Small セットを用いて，バッチサイズ 100，エポック数 1 とし，4 枚の NVIDIA Tesla V100 GPU により学習した．最適化には Adam [13] を用い，$\beta_1 = 0.9$，$\beta_2 = 0.999$，$\epsilon = 10^{-8}$ とした．モデルパラメータは 0 で初期化したバイアスを除き，$N(0, 0.02)$ で初期化した．学習率は 1×10^{-6} とし，学習の半分までは 0 から線形で増加させ，最終ステップで 9×10^{-7} になるように線形に減衰させた．勾配は最大ノルム 1 でクリッピングした．BERT は base モデル（768 次元）を利用し，2 層の出力層の中間層の次元数は 1000，最終層（スパースベクトル v）の次元数 D は 30000 とした．hinge loss のマージン ϵ は 1.0 とした．トークナイズには BERT Wordpiece トークナイザ[1)]（語彙数 30K）を用いた．

表 2 MS MARCO 2.1 の passage ranking のデータの詳細.

Set	Num records
Passage	8,841,823
Train Triples Small	39,782,779
Train Triples Large	397,756,691
Dev. queries	6980

top α 学習のためのハイパーパラメータとして $\alpha_{\mathrm{q}} = 0.1\%$ とした．また，バッチサイズ 100 とした場合，バッチ内で上位 α% を計算すると 閾値がバッチ内のデータに強く依存しすぎるため，計算を安定させるために現在のステップを含め過去 20 ステップ分の訓練データのベクトルを保存しておき，これらを合わせて用いることで top α の閾値を安定させた．ここで，現在のステップを除く 19 ステップ分のデータについては閾値の決定にのみ用い，計算グラフは保持せずパラメータの更新は行わないものとした．

1) https://huggingface.co/transformers/model_doc/bert.html#berttokenizer