# オープンドメイン QA における DPR の有効性検証

加藤 拓真 [1*] 宮脇 峻平 [1*] 西田 京介 [2] 鈴木 潤 [1,3]
[1] 東北大学 [2]NTT メディアインテリジェンス研究所 [3] 理化学研究所
[1]{takuma.kato, miyawaki.shumpei, jun.suzuki}@ecei.tohoku.ac.jp
[2]kyosuke.nishida.rx@hco.ntt.co.jp

## 1 はじめに

計算機による自然言語の適切な理解を実現することは，自然言語処理および人工知能研究の最終目的の一つである．その実現に向けた取り組みとして，質問応答（QA）タスク に関する研究が近年大きな注目を集めている．QA タスクでは，質問に関連する適切な知識に基づいて解答を推定することが求められるが，関連知識を含む文書集合が用意されないという状況は少なくない．このような状況に対して，オープンドメイン QA [1] は，問題を解くために必要な知識源を特に規定しない QA タスクとして，非常に重要な役割を持つ．

一般的にオープンドメイン QA では，1) 検索モジュール（*retriever*）が，質問に対して関連する文書を大規模な文書集合から検索し，2) 読解モジュール（*reader*）が，検索した関連文書を用いて質問の解答を推定する，二つのプロセスを用いて取り組まれることが多い．特に関連文書の検索性能が，QA システムの性能に与える影響は大きく [1]，Karpukhin ら [2] は，ニューラルモデルによる文書エンコーダによって密なベクトル表現にエンコードされた意味的な情報を，関連文書の検索に用いる Dense Passage Retriever（DPR） を提案し，QA システムの性能改善を実現した．DPR を用いたオープンドメイン QA に対する取り組みは，今後更なる発展が期待されているが，DPR の詳細な実験設定がオープンドメイン QA に与える影響を調査した研究は少なく，DPR について明らかにされていない要素が多い．そこで本研究では，DPR の応用的な利用を目的として，特に次元数を変化させたときの検索時間や性能変化について定量的な評価および分析を行う．

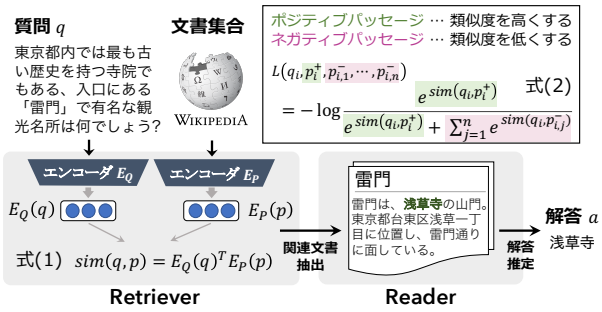


**図 1 DPR を用いたオープンドメイン QA システムの概要図．** 1) 質問に対する関連文書を抽出するため，エンコードされた質問と文書の類似度を計算する *retriever* と，2) *retriever* が抽出した関連文書から質問に対する解答の推定を行う *reader* の二つのモジュールで構成される．

## 2 関連研究

オープンドメイン QA では，モデルの予測性能に大きな影響を与える関連文書の検索方法を対象に，様々な研究が行われてきた．従来，関連文書の検索方法には TF-IDF や BM25 [3] のように，質問中に存在する単語に対する文書中の出現頻度などの統計量を用いたマッチングが使用された．しかし QA タスクでは，質問文の表現と検索対象に記載される情報の表記が必ずしも同じであるとは限らないため，表層一致を仮定する BM25 や TF-IDF といった抽出方法では，適切な文書が検索できない可能性が高いことが問題として広く知られている．この問題の解決策として，近年急速に発達した単語埋め込みや深層ニューラルネットに基づく意味的な情報を含む密なベクトル表現を用いて，質問に対して意味的に類似する文書を抽出する方法が挙げられる．意味的な情報を含む密なベクトル表現を用いることで，質問中に含まれる単語の出現頻度に基づくマッチングだけでは抽出できなかった，未知の情報に関する関連文書の取得が可能となる．密なベクトル表現を用いた関連文書の抽出方法は様々なタスクに対して研究されてきており [4][5][6]，中でも Karpukhi らは

* 第一著者と第二著者の貢献は同等である．

DPR という手法を提案し，密なベクトル表現によって QA システムの性能が向上したことを報告している．*retriever* について調査している先行研究として，Luan ら [7] は密なベクトル表現の次元数についての調査を行なっている．Luan らの研究では，密なベクトル表現 の次元数がオープンドメイン QA に与える影響が明らかになっていない点と，DPR を研究対象としていない点が，我々の研究と異なっている．

# 3 DPR を用いたオープンドメイン QA システム

1 章で述べたように，オープンドメイン QA システムは，文書検索を行う *retriever* と検索した文書から解答を推定する *reader* の二つのモジュールで構成される．本研究では，*retriever* における関連文書の検索手法として Dense Passage Retriever （DPR） [2] を用いた QA システムを使用する (概要を図 1 に示す)．本章では，この DPR を用いた QA システムについて説明する．

## 3.1 retriever による文書検索

本節では，*retriever* として DPR を用いた文書検索について説明する．

### 3.1.1 DPR の学習

DPR では与えられた質問 $q$ および文書 $p$ の類似度を計算し，質問に関連する文書の類似度が高くなるように学習を行う．類似度は，質問 $q$ と文書 $p$ との内積によって計算される（式 1）．

$$sim(q,p) = E_Q(q)^\top E_P(p) \tag{1}$$

ここで，$E_Q$，$E_P$ はそれぞれ質問，文書に対する二つのエンコーダを表す．DPR では，これら二つのエンコーダについて学習を行う．学習データは，$D = \{\langle q_i, p_i^+, p_{i,1}^-, \cdots, p_{i,n}^-\rangle\}$ で定義される．ここで，$p_i^+$ は正例の文書，$p_{i,j}^-$ は負例の文書を示す．モデルは式 2 の損失関数が最小になるようにエンコーダ $E_Q$ と $E_P$ を学習する．

$$L(q_i, p_i^+, p_{i,1}^-, \cdots, p_{i,n}^-) = -\log \frac{e^{sim(q_i,p_i^+)}}{e^{sim(q_i,p_i^+)} + \sum_{j=1}^{n} e^{sim(q_i,p_{i,j}^-)}} \tag{2}$$

各質問に対して，式 2 で求められる損失は，正例文書との類似度が高いほど，また負例文書との類似度が低いほど，値が小さくなる．3.1.2 項では，正例および負例となる文書の選択方法について説明する．

### 3.1.2 正例・負例の選択

本項では，3.1.1 項で述べた，正例および負例の文書について，その選択方法を述べる．以下，正例文書をポジティブパッセージとする．また，負例文書をネガティブパッセージおよびハードネガティブパッセージの二つに分ける．

**ポジティブパッセージ**　文書集合のうち，与えられた質問に対する解答が文書中にも含まれ，かつ，BM25 スコアによる質問と文書間の関連度の高い上位 100 件に含まれる文書をポジティブパッセージとして定義する．なお上位 100 件以内に解答が含まれない場合，その質問解答ペアはデータセットとして使用しないものとする．

**ネガティブパッセージ**　質問 $q_i$ に対して，同一ミニバッチ内に存在する他の質問解答ペアが保持するポジティブパッセージを，質問 $q_i$ に対するネガティブパッセージとして定義する[1)]．

**ハードネガティブパッセージ**　質問に対する解答が含まれない 文書集合のうち，質問との関連度が最も高い文書をハードネガティブパッセージとして定義する．

## 3.2 reader による解答推定

本節では，*reader* の解答推定について説明する．*reader* では，*retriever* が検索した関連文書から問題の解答となる箇所を推定する．*retriever* が検索した上位 $k$ 件の文書を，BERT [8] でエンコードし，分散表現 $\mathbf{P}_i \in \mathbb{R}^{L\times h}$ $(1 \le i \le k)$ を獲得する．ここで，$L$ は文書の最大長，$h$ は BERT の隠れ層の次元数である．解答を推定するための文書を決定するため，獲得した分散表現に対して，式 3 を用いて各文書のスコアを求める．また，$i$ 番目の文書について，文書中の位置 $s$ のトークンが解答スパンの開始トークンになる確率 $P_{\text{start},i}$ と，位置 $t$ のトークンが解答スパンの終了トークンになる確率 $P_{\text{start},i}$ を，それぞれ式 4,5 で求める．

$$P_{\text{selected}}(i) = \text{softmax}(\hat{\mathbf{P}}^\top \mathbf{w}_{\text{selected}}))_i \tag{3}$$

$$P_{\text{start},i}(s) = \text{softmax}(\mathbf{P}_i \mathbf{w}_{\text{start}})_s \tag{4}$$

$$P_{\text{end},i}(t) = \text{softmax}(\mathbf{P}_i \mathbf{w}_{\text{end}})_t \tag{5}$$

ここで，$\mathbf{P} = [\mathbf{P}_1^{[\text{CLS}]}, ...\mathbf{P}_k^{[\text{CLS}]}]$ ([CLS] は BERT の [CLS] トークン) であり，$\mathbf{w}_{\text{selected}}, \mathbf{w}_{\text{start}}, \mathbf{w}_{\text{end}} \in \mathbb{R}^h$ は，*reader* が学習する重みベクトルである．また，softmax

1） 訓練時にはランダムにミニバッチの作成が行われる度，プログラム上で動的にネガティブパッセージの作成を行う．

表 1　データセットの質問数

| | 訓練 | 開発 | 評価 |
|---|---|---|---|
| 前処理前 | 13,061 | 995 | 997 |
| 前処理後 | 5,131 | 379 | 406 |

表 2　ハイパーパラメータ

| | retriever | reader |
|---|---|---|
| Seed | 3 | 3 |
| Batch Size | 16 | 8 |
| Learning Rate | $1.0 \times 10^{-5}$ | $2.0 \times 10^{-5}$ |
| Warmup Steps | 1237 | 1237 |
| Gradient Accumulation Steps | 3 | 1 |
| Dropout Ratio | 0.3 | 0.2 |
| Optimizer | Adam | Adam |

の添字 $i,s,t$ は，それぞれ softmax の結果から添字 $i,s,t$ の行を取得するという意味である．最終的に，式 3 で求めたスコアが最も高い文書について，$P_{\text{start},i}(s) \times P_{\text{end},i}(t)$ の値が最大となるスパン $(s,t)$ を，*reader* が予測結果として出力する．

# 4　実験設定

## 4.1　データセット

本実験では，QA タスクの日本語データセットとして JAQKET [9] を用いた．また，検索対象の文書となるオープンソースの文書集合として，920,172 の記事からなる日本語 Wikiepdia のデータを用いた．

## 4.2　データセットの前処理

MeCab [10] を使用して Wikipedia の 記事をトークナイズし，各文書のトークン数が 200 となるように各記事を分割した[2]．トークナイズによる文書の作成をにより，合計で 6,125,433 からなる文書集合を取得した．また 3.1.2 項における正例および負例の文書の選択は，ElasticSearch[3] を用いて質問と Wikipedia の文書間の関連度となる BM25 スコアを算出し，求めた関連度に基づいて選択を行った．前処理を行ったデータセットの統計情報を表 1 に示す[4]．

2）　トークン数は，$\{50, 100, 150, 200\}$ から，開発セットにおける順位平均値が最も低い値を示した 200 を選択した．
3）　https://github.com/elastic/elasticsearch
4）　前処理の前後でデータサイズが減少しているが，これは 3.1.2 項で述べたように，質問と解答が含まれる文書との関連度が上位 100 件に含まれない場合，その質問はデータセットとして使用しないことが原因である．

## 4.3　モデルアーキテクチャ

本実験で用いる QA システムは，3 章で述べた，*retriever* と *reader* の二つのモジュールから構成されるモデルを使用した．エンコーダには *retriever*, *reader* 共に事前訓練済みの日本語 BERT[5] を用いた．表 2 に *retriever* および *reader* のハイパーパラメータの値を示す[6]．*retriever* による関連文書の抽出には，質問および文書に関する二つのエンコーダから取得した密なベクトル表現を用いて，式 1 の内積値に基づく関連度を算出した．関連度の計算ツールには，検索時間の効率化を図るため，faiss [7] を使用した．faiss の検索手法には，最大内積探索に基づいた IndexFlatIP を用いて，関連文書の検索を行った．なお全ての文書のエンコードに要する時間は，TITAN X（Pascal）を用いた場合，およそ 5 時間 25 分ほどであった．*reader* の性能を測る評価指標として，質問に対する正解の解答との完全一致を用いた．

# 5　実験

本実験では，DPR の要素の 1 つである密なベクトル表現の次元数が QA システムの性能に与える影響を明らかにするために，ベクトルの次元数を変えたときの性能評価を行った．*retriever* の性能評価として， 5.1 節では，faiss を用いて関連文書の検索を行った際の検索時間の結果を，また 5.2 節では，抽出した関連文書の評価結果を示す．さらに 5.3 節では，質問に対する QA システムの正解率の結果を示す．

## 5.1　関連文書の検索時間

*retriever* の文書エンコーダを用いて，Wikipedia の全文書のエンコードを行い，faiss を用いた関連文書の検索を行った．密なベクトル表現の各次元に対して，faiss のインデックスの構築時間，および関連文書の検索時間を表 3 に示す．なお GPU は TITAN X（Pascal）を使用した．また関連文書は上位 300 件までを対象に抽出を行った．実験結果より，関連文書の検索に要する時間は，検索に用いる質問および文書におけるベクトル表現の次元数におおよそ比例することが分かった．これは関連文書の検索に用いた

5）　https://github.com/cl-tohoku/bert-japanese
6）　開発セットを用いて，*retriever* では順位平均値が最小となった値，*reader* では完全一致した数が最多となった値を，設定値とした．
7）　https://github.com/facebookresearch/faiss

表 3　faiss によるインデックス構築時間および検索時間

| 次元数 | 768 | 384 | 192 | 96 |
|---|---|---|---|---|
| インデックス構築時間 [sec] | 100 | 54 | 32 | 14 |
| 一問あたりの検索時間 [msec] | 99 | 52 | 29 | 16 |

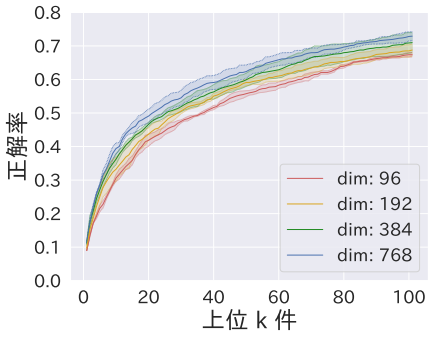


図 2　開発セットにおける *retriever* の正解率

手法が，式 1 に示す内積計算に基づいたもので，ベクトル表現の次元数に比例して，演算の計算量が増加するためであると考えられる．

## 5.2　retriever の性能評価

各質問に対して，*retriever* が抽出した上位 $k$ 件までの文書中に解答が含まれる場合を正解とし，*retriever* の性能評価を行った．評価結果を図 2 に示す．図 2 より，密なベクトル表現の次元数が大きいほど，解答を含む文書の抽出漏れが少ないことを確認した．しかし，上位 100 件までの文書集合でさえ解答が含まれる割合は七割程度であり，残り三割程度の質問は解答不可能であることを示している．

## 5.3　QA システムの性能評価

*retriever* が抽出した関連文書の集合のうち，上位 $k$ 件までの文書を使用した時の正解率を評価指標として，QA システムの性能評価を行った．評価結果を図 3 に示す．図 3 より，密なベクトル表現の次元数が最も大きい 768 次元での正解率が，最も高い値となった．また，与えられた質問に対して，*reader* が予測した解答のスパンに対して，正解のスパンと部分的に一致している事例がどれほど存在しているか調査を行った．具体的には，式 6 に示す Intersection over Union の値を算出し，$\lambda$ の値を超えた場合の予測事例を正解と判定した場合の正解率を算出した．

$$\frac{T_{gold} \cap T_{pred}}{T_{gold} \cup T_{pred}} \geq \lambda \tag{6}$$

ここで，$T_{gold}$，$T_{pred}$ は，正解の解答およびモデルが予測した解答を，それぞれ MeCab でトークン単位に分割した集合を表す．$\lambda$ はモデルの予測が正解とどの程度一致しているかを定めるハイパーパラメータである．今回は抽出された上位 100 件の文書を対象として，*reader* の予測に対する正解率を，$\lambda = 1.0$ と $\lambda = 0.5$ それぞれについて算出した．評価結果を表 4 に示す．表 4 より，$\lambda = 1.0$ および $\lambda = 0.5$ では性能の差がほとんどみられなかった．このことから，モデルの予測が誤っている場合，その予測スパンは，正解の解答スパンとは異なった部分を文書から抜き出しているということが考えられる．

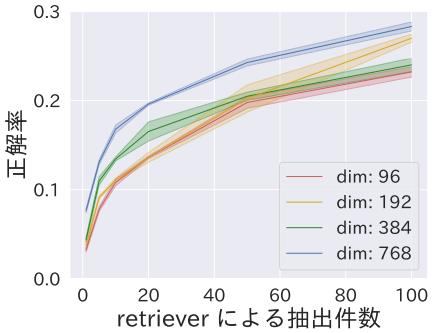


図 3　開発セットにおける QA システムの正解率

表 4　密ベクトルの次元数に対する QA システムの正解率

| 次元数 | $\lambda$ | 開発 | 評価 |
|---|---|---|---|
| 768 | 1.0 | 28.32 ± 0.50 | 28.16 ± 0.23 |
| | 0.5 | 28.76 ± 0.57 | 28.41 ± 0.23 |
| 384 | 1.0 | 24.01 ± 0.75 | 24.96 ± 1.82 |
| | 0.5 | 24.54 ± 0.75 | 24.96 ± 1.82 |
| 192 | 1.0 | 27.00 ± 0.45 | 24.71 ± 1.29 |
| | 0.5 | 27.26 ± 0.33 | 24.71 ± 1.29 |
| 96 | 1.0 | 23.22 ± 0.57 | 24.79 ± 1.23 |
| | 0.5 | 23.39 ± 0.69 | 25.12 ± 1.26 |

# 6　おわりに

本研究では，DPR を用いた密なベクトル表現が，QA システムの性能にどのような影響を与えるか調査するため，JAQKET データセットを用いて，QA システムの予測性能や関連文書の検索時間に与える影響を評価した．結果として，密なベクトル表現の次元数が文書の検索時間やモデルの性能に与える影響が明らかになった．本実験では，一つのデータセットに対象を限定して実験を行ったが，今後は複数のデータセットに対して同様の実験を行い，データ間での横断的な比較について調査を行っていきたい．

**謝辞**　本研究の一部は JSPS 科研費 JP19H04425 の助成を受けたものである．

# 参考文献


[1] Ellen M. Voorhees. The TREC-8 question answering track report. In *TREC*, 1999.

[2] 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*, 2020.

[3] S. Robertson and H. Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, Vol. 3, , 2009.

[4] Wen-tau Yih, Kristina Toutanova, John C. Platt, and Christopher Meek. Learning discriminative projections for text similarity measures. In *CoNLL*, 2011.

[5] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry P. Heck. Learning deep structured semantic models for web search using clickthrough data. In *ACM CIKM*, 2013.

[6] Daniel Gillick, Sayali Kulkarni, Larry Lansing, Alessandro Presta, Jason Baldridge, Eugene Ie, and Diego García-Olano. Learning dense representations for entity retrieval. In *CoNLL*, 2019.

[7] Yi Luan, Jacob Eisenstein, Kristina Toutanova, and Michael Collins. Sparse, dense, and attentional representations for text retrieval. *CoRR*, Vol. abs/2005.00181, , 2020.

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

[9] 鈴木正敏, 鈴木潤, 松田耕史, 西田京介, 井之上直也. JAQKET: クイズを題材にした日本語 QA データセットの構築. 言語処理学会, 2020.

[10] Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. Applying conditional random fields to japanese morphological analysis. In *EMNLP*, 2004.