CE-002

疑似的演繹推論を行う質問応答システムの開発
Development of a Question Answering System That Utilizes Pseudo Deductive Reasoning

田中 一晶† 髙橋 ともみ† 小林 賢一郎‡ 中谷 宙詩‡ 岡 夏樹†
Kazuaki Tanaka Tomomi Takahashi Kenichiro Kobayashi Hiroshi Nakatani Natsuki Oka

1. はじめに

1.1 質問応答システム

質問応答システムは古くから研究されており，テキストや音声で入力されたユーザからの質問に対し，対話形式で答えを返すものである．近年では，スマートフォン，AI スピーカをはじめとする様々な電子機器に実装されている音声アシスタントの代表的な機能の 1 つとなっている．音声入力で気軽に必要な情報が得られることから，情報検索の手段として期待されている．

質問応答システムは基本的にユーザと 1 問 1 答形式で対話するものであるが，広くは，マルチターンの会話も含む対話システムや雑談システム（Chatbot）に類するシステムと言える．そのため，対話システムと同様に， Close Domain か Open Domain かという側面と，システムの返答が Retrieval（検索）か Generative（生成）かという側面から分類することができる[1]．Close Domain の場合，ユーザからの入力は特定の話題に限定されるが，Open Domain の場合は話題が限定されない．その入力に対し，適切な返答を予め用意した文や単語から選択するのが Retrieval モデルであり，機械学習技術で生成するのが Generative モデルである．Retrieval モデルはユーザからの入力が予想しやすい Close Domain に限られるが，予め用意した文を使用するため，不自然な返答になりにくいという利点がある．一方，Open Domain な入力に対しては Generative モデルでしか返答できず，Retrieval モデルでは返答することは困難であるとされてきた．

質問応答タスクの SOTA を達成した機械学習技術である BERT では，文章中から入力文に対応する答えを抽出することができる[2]．しかしながら，答えを含む Passage と呼ばれる文章は人手で与える必要があり，その答えは文章に含まれる単語や短いフレーズをそのまま返すため，分類としては Close Domain であり Retrieval モデルと言える．これに対し，質問に対して文で返答する Generative モデルが提案されている[3]．基本的に文を生成する Generative モデルの学習には大量の文章データセットを必要とし，その性能は学習データのドメインにおいて評価されている．したがって，回答に Passage を必要とするだけでなく，性能が発揮できるドメインが学習データに依存する面からも Close Domain と言える．

BERT による質問応答を Open Domain 化する取り組みも行われている[4][5]．これは，入力文と Wikipedia の記事を自動的に対応付ける仕組みを加えたもので，対応する記事を Passage とすることで Open Domain 化している．BERT の学習済みモデル自体の学習に Wikipedia が使用されていることから，Wikipedia の記事を Passage として使用することと相性が良い．答えが含まれると推測される文章を入力文からまず検索する仕組みは Open Domain な Retrieval モデルを構築する上で有効な手段と考えられるが，Wikipedia 以外のデータベースに対しても高い性能を発揮できるかは不明である．

† 京都工芸繊維大学 Kyoto Institute of Technology
‡ TIS 株式会社 TIS Inc.

1.2 疑似的な演繹推論によって返答する提案システム

本研究では，機械学習モデルのデータセットとして十分な量が得られないデータベースにも適用可能な質問応答システムを提案する．その質問応答の流れは次の通りである．まず，質問応答のための文章から，条件部と帰結部を抽出する[6]．本研究では，これを推論データベースと呼ぶ．例えば，「海外旅行に行く場合はパスポートが必要です」という文ならば，「海外旅行に行く場合は」を条件部，「パスポートが必要です」を帰結部として推論データベースに保存する．そして，入力文に類似する条件部を検索し，類似する条件部に対応する帰結部を返答とする．例えば，「海外旅行に行くんだ」という入力に対して，上記の例では「海外旅行に行く場合は」という条件部との類似度が高くなり，システムは「パスポートが必要です」と返答する．これにより，あたかもシステムが演繹推論を行っているかのように返答することができる．もちろん，提案システムによって全ての入力に対応できるわけではないが，学習データセットとして十分な量が得られないドメインの知識を利用した返答が可能になることや，ユーザからの入力が明確な質問文でなくても，演繹推論によって能動的にユーザに知識が与えられるという 2 つの利点がある．入力文から検索すべきデータベースを決定する仕組みや[4][5]，先行研究で提案された既存の質問応答システムに，本研究の提案システムを組み合わせることで，返答の多様性を高められる可能性がある．

入力文と条件部の類似文検索には，Sentence BERT（以下，SBERT と呼ぶ）を使用する．これば BERT の埋め込み表現（文の意味を表現するベクトル）を改善したもので，類似文検索において高い性能を発揮する[7]．SBERT は学習済みの BERT モデルを類似文のデータセットを用いてファインチューニング（再学習）することで構築する．これも先に述べた機械学習モデルと同様に，大量の学習データを必要とする問題と，その性能が学習データセットのドメインに依存するという問題がある．そこで本研究では，性質が大きく異なる 2 つのドメインの学習データセットを混合してファインチューニングすることで，学習データセットに含まれない未知のドメインに対する類似文検索の性能向上が図れると考え，どのように混合することが性能向上に繋がるか検証する．

2. 関連研究

近年の機械学習技術を用いた質問応答システムの多くは BERT[2]を使用している．BERT を使用することで埋め込み表現と呼ばれる文の意味を表現するベクトルを得ることができ，学習済みの BERT モデルをファインチューニング

することで質問応答だけでなく雑談対話等の様々な言語タスクに応用することができる．質問応答においては，質問文とその質問に答えるのに必要な情報を含む Passage と呼ばれる文章のデータセットでファインチューニングを行うことで，高い正解率で読解タスクが行えることが示されている．その性能や機能を向上させるため，Passage を自動選択してオープンドメイン化する手法[4][5]，プルーニング（刈り込み）と呼ばれるネットワーク構造の最適化[8]，マルチターンの質問応答[9]，複数段落の Passage からの答えの推測等[10]，盛んに研究されている．これらの研究では，いずれも入力文は 5W1H の明確な質問文である必要があり，Passage に質問の答えとなる情報が含まれていなければ回答不能となる．

本研究が提案する質問応答システムは，入力文と推論データベース中の文との類似文検索によって，明確な質問文でなくても応答することが可能である．したがって，入力が明確な質問文の場合には既存の質問応答システムで適切な答えを，そうでない場合は提案システムによって関連する知識をユーザに与える，より多様な入力に対応した質問応答システムを構築できる可能性がある．

類似文検索を用いる質問応答システムは先行研究でも提案されており，その多くは質問サイトの QA データセットを使用して BERT をファインチューニングし，その類似文検索モデルを使用して入力文と類似する悩み[11]や QA ペア[12]をデータベースから検索するものである．そして，その類似文検索を木構造で高速化した質問応答システムも提案されている[13]．これらのシステムも類似文検索を用いるため，我々の提案システムと同様に，入力文が明確な質問文でなくても関連する情報をユーザに与えられると思われる．しかしながら，その類似文検索モデルの構築には，検索対象であるデータベースと同じドメインのデータセットを使用しているため，学習データとして十分な数が得られない未知のドメインのデータベースに適用できるかは不明である．

本研究では，類似文検索用の SBERT を構築する上で，性質が異なる 2 つのデータセットを混合することで，それらのドメインとは異なる未知のドメインのデータベースに対する類似文検索の性能の向上を図る．また，本研究で検索対象とする推論データベースでは，QA ペアではなく条件部と帰結部のペアで構成されており，入力文と類似する条件部に対応する帰結部を返答とすることで，演繹推論によってユーザの意図を汲み取った返答が行える可能性がある．次章では，どのようにデータセットを混合することが未知のドメインのデータベースに対する類似文検索の性能を向上させるか検証する．

3. 類似文検索のための SBERT 構築

BERT のファインチューニングによって類似文検索用の SBERT を作成する上で，どのように学習データを作成するかが課題となる．具体的には，類似した文のセットを大量に用意する必要がある．その方法の 1 つとして，画像に対して複数の日本語のキャプションがつけられたデータセット（STAIR Captions[14]）を使用する方法が提案されている[15]．これは，同じ画像につけられたキャプションは類似しているという仮説に基づいて，同じ画像につけられた 2 つのキャプションを類似文のセットとして使用する方法である．また別の方法としては，Wikipedia の同じ見出しから抽出した 2 文を類似文として使用する方法も提案されている[7]．これらの手法はそれぞれに利点と欠点があることが予想される．まず，画像のキャプションを利用する方法では，類似文同士の意味が非常に近く，質の高い類似文のデータセットであると考えられるが，キャプションは文としては短いため，ファインチューニングにおいては文の意味としての類似性よりも単語レベルでの類似性が重視される可能性がある．一方，Wikipedia を使用する方法では，同じ段落であっても繰り返し同じ内容の文が出現することは稀であり，ファインチューニングにおいては話題レベルの広い類似性が学習される可能性がある．そこで，本研究では，これらの方法で作成したデータセットを混合したデータセットを使用してファインチューニングすることで，単語レベルから話題レベルまで多様な類似性を考慮した類似文検索が行える SBERT を構築できると考えた．この仮説を検証するため，いずれか一方の方法で作成したデータセットでファインチューニングした SBERT と，混合したデータで作成した SBERT の性能を比較する実験を行う．また，混合する方法として，データ数を等しくする必要があるか検証するため，アンバランスなデータとアンダーサンプリングしたデータでそれぞれ作成した SBERT の性能を比較する実験も行う．これらの実験でファインチューニングする BERT は，NICT BERT 日本語 Pre-trained モデル（BPE あり）を使用した[16]．以下では，それぞれの方法の詳細な内容と比較結果について説明する．

3.1 STAIR Captions によるファインチューニング

STAIR Captions では約 12 万の画像にそれぞれ 5 つの日本語のキャプションが付与されている．その例を図 1 に示す．この例の通り，付与されたキャプションは概ね類似していることが見受けられるが，画像中のどの部分に着目したキャプションであるかによって類似していないキャプションのペアも含まれている．そこで，キャプションの中から類似文のペアを選出する上で，意味が大きく外れたものを省くため，5 つのキャプションを GiNZA[1]でベクトル化し，COS 類似度が閾値以上のものを選出した．具体的には，まずキャプションの中から 1 つをアンカー文として選出し，それに対して COS 類似度が 0.85 以上のキャプションを類似した文（ポシティブ文）として選出する．そして，他の画像につけられた 1 つの文を類似していない文（ネガティブ文）として選出する．以上の 3 つの文（アンカー文，ポジティブ文，ネガティブ文）のセットを 1 つのデータ（Triplet データ）とする．この操作を行った結果，約 12 万のデータが作成され，これを 8 : 1 : 1 の割合で学習データ（98,252），検証データ（12,282），テストデータ（12,282）に分割した．

作成した学習データを用いて BERT のファインチューニング（バッチサイズ 16）を行い，検証データで正解率を調べた結果，1 回のエポックで 0.993 の正解率が得られた．この結果から，学習回数は十分であると判断した．テストデータは他の SBERT との比較評価の際に使用する．キャ

[1] GiNZA - Japanese NLP Library
https://megagonlabs.github.io/ginza/
（参照 2022-6-17）

図 1　STAIR Captions に含まれる画像とキャプションの例

プションデータでファインチューニングした SBERT を以下，**Cap** と呼ぶ．

3.2 Wikipedia によるファインチューニング

Wikipedia データベースは約 3GB の容量があり，扱いやすいように専用のソフトフェア（WikiExtractor[2]）を利用してテキストファイルに変換した．その結果，AA～BF という名前のフォルダ 32 個の各フォルダ内に wiki_00～wiki_99 の 100 個のテキストファイル（最後の BF フォルダは 32 個）が作成される．各テキストファイルには約 3000 文が含まれている．

この Wikipedia データベースを使用してファインチューニングを行う．このデータには Wikipedia の本文の他に，タイトルや見出しが含まれている．このうち，1 つの文をアンカー文とし，それと同じタイトルで同じ見出しの文をポシティブ文として選出する方法が SBERT の論文で提案されている[7]．まず，学習データと同じ見出しの文が検証データやテストデータに含まれることを避けるため，予め Wikipedia データベースの 32 個のフォルダを，24 : 4 : 4 に分割し，それらから学習データ，検証データ，テストデータをそれぞれ作成した．

ポジティブの選出において，ランダムに選出する方法と，キャプションデータを使用したファインチューニング（3.1 節）と同様に類似度が高い文を選出する方法が考えられる．その際には，GiNZA によって文をベクトル化して類似度を比較したが，そのベクトルは，文に含まれる単語のベクトルの平均であるため，文としての意味はほとんど考慮されていない．したがって，Wikipedia に含まれる長い文に対しては適切に類似文を選択できない可能性が考えられる．そこで，3.1 節で構築した SBERT（Cap）を用いて文のベクトルを作成し，同じ見出しの文の中で最も COS 類似度が高く，その類似度が 0.98 より高い文をポシティブ文とした．そして，他のタイトルから選出した文をネガティブ文とするが，類似度が 0.80 未満のものを選出した．その結果，学習データ（549,352），検証データ（91,638），テストデータ（72,042）が得られた．

次に，同じタイトル同じ見出しの文からランダムにポジティブ文を選出する方法でもファインチューニングを行った．また，同じタイトル同じ見出しの文からあえて類似度が低い文をポシティブ文として選出することで，類似度が高い文を選出する場合や，ランダムに選出する場合よりも明確に性能が劣化するのか確認するため，アンカー文に対するポシティブ文の類似度が 0.90 未満であり且つ，ネガティブ文の方がポシティブ文よりも類似度が高いデータも作成しファインチューニングを行った．これらの方法では，作成されるデータ数が非常に多くなるため，データ数を統制するため，類似度が高い文を選出した場合と同じ量のデータ（学習データ（549,352），検証データ（91,638），テストデータ（72,042））をランダムに抽出した．

作成した 3 種類の Triplet データを用いて BERT のファインチューニング（バッチサイズ 16）を行い，検証データで正解率を調べた結果，1 回のエポックで十分に高い正解率が得られた（類似度が高い文を選出したものと混合：0.999，ランダムに選出：0.985，類似度が低い文を選出：0.983）．したがって，キャプションデータでのファインチューニングと同様に学習回数は十分であると判断できる．以下では，Wikipedia データベースで構築した 3 つの SBERT を次のように呼ぶ．

Wiki_near：　類似度が高い文をポジティブ文として選出
Wiki_rand：　ランダムにポジティブ文を選出
Wiki_far：　類似度が低い文をポジティブ文として選出

3.3 混合データによるファインチューニング

2 種類の学習データを混合してファインチューニングを行う効果を確認するため，STAIR Captions から作成した Triplet データ（3.1 節）に，Wikipedia データベースから作成した 3 種類の Triplet データ（3.2 節）をそれぞれ混合した Triplet データ（学習データ（647,604），検証データ（103,920））でファインチューニングを行った．検証データで正解率を調べた結果，混合した場合でも 1 回のエポックで十分に高い正解率が得られた（類似度が高い文の選出：0.998，ランダムに選出：0.990，類似度が低い文を選出：0.985）の正解率が得られた．これによって構築した 3 つの SBERT を次のように呼ぶ．

Mix_near：　類似度が高い文をポジティブ文として選出した Triplet データと混合
Mix_rand：　ランダムにポジティブ文を選出した Triplet データと混合
Mix_far：　類似度が低い文をポジティブ文として選出した Triplet データと混合

3.4 アンダーサンプリングして混合したデータによるファインチューニング

まず，Wikipedia から作成した 3 種類の Triplet データの数をアンダーサンプリングし，STAIR Captions から作成した Triplet データの数に揃えた Triplet データ（学習データ（98,252），検証データ（12,282））を作成した．さらに，それらの Triplet データと STAIR Captions から作成した Triplet データを混合した Triplet データ（学習データ（196,504），検証データ（24,564））を作成した．

これら 6 つの Triplet データでファインチューニングを行った．混合した Triplet データだけでなく，データ数を揃えただけの Triplet データでも SBERT を構築したのは，アンダーサンプリングによってデータ数が減少することによる効果を確認するためである．検証データで正解率を調べた結果，上述と同様に 1 回のエポックで十分に高い正解率が得られた（類似度が高い文を選出：0.999，類似度が高い文を選出したものと混合：0.995，ランダムに選出：0.988，

[2] WikiExtractor
https://github.com/attardi/wikiextractor
（参照 2022-6-17）

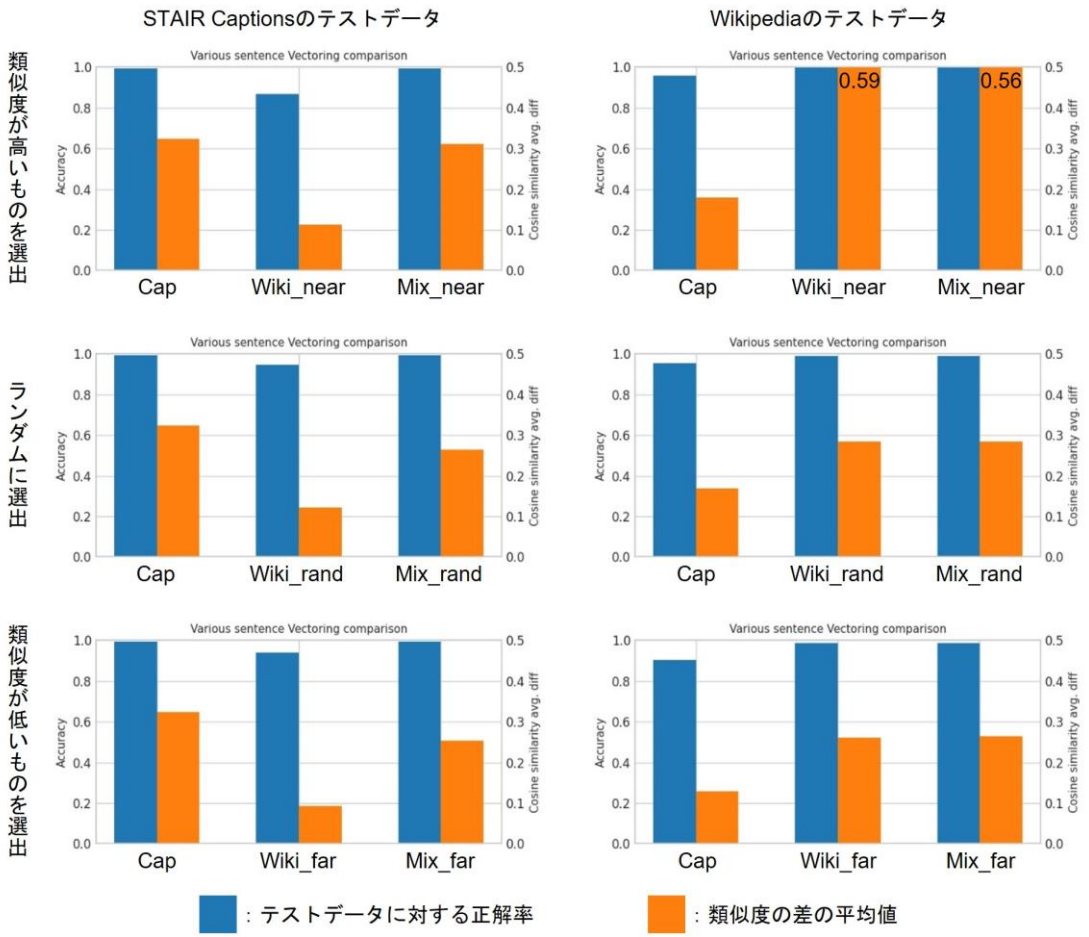

図 2　アンダーサンプリングなしの学習データでファインチューニングした SBERT の性能比較

ランダムに選出したものと混合：0.989，類似度が低い文を選出：0.980，類似度が低い文を選出したものと混合：0.985）．これによって構築した 6 つの SBERT を次のように呼ぶ．

Wiki_near_us： 類似度が高い文をポジティブ文として選出したデータからアンダーサンプリング

Wiki_rand_us： ランダムにポジティブ文を選出した Triplet データからアンダーサンプリング

Wiki_far_us： 類似度が低い文をポジティブ文として選出した Triplet データからアンダーサンプリング

Mix_near_us： 類似度が高い文をポジティブ文として選出した Triplet データからアンダーサンプリングして混合

Mix_rand_us： ランダムにポジティブ文を選出した Triplet データからアンダーサンプリングして混合

Mix_far_us： 類似度が低い文をポジティブ文として選出した Triplet データからアンダーサンプリングして混合

4. テストデータによる SBERT の性能比較

3.1～3.3 節で構築した 13 の SBERT の性能を，STAIR Captions から作成したテストデータと Wikipedia から作成したテストデータで比較した．その結果を図 2 と図 3 に示す．図 2 は STAIR Captions から作成した Triplet データと Wikipedia から作成した Triplet データの数を揃えていないアンダーサンプリングなしの SBERT を比較した結果であり，図 3 はアンダーサンプリングありの SBERT を比較した結果である．各 SBERT においてテストデータをベクトル化し，アンカー文に対する COS 類似度が，ネガティブ文よりもポジティブ文の方が高ければ正解とし，その正解率を青の棒グラフに示している．また，ポジティブ文との類似度がネガティブ文との類似度よりも高ければ高いほど明確に類似した文を検出できるモデルであるため，それらの類似度の差の平均値（ポジティブ文との類似度－ネガティブ文との類似度／テストデータ数）も評価した．その結果を橙の棒グラフが示している．

全ての比較結果に共通する事柄として，ファインチューニングに使用したドメインのテストデータに対して 100%近い正解率を示しており，SBERT 自体が適切に文の意味を捉えてベクトル化できていることが考えられる．エポック数が 1 回で十分であったことからも，ベースとして使用した BERT の性能が非常に高いことが考えられる．一方，ファインチューニングに使用していないドメインのテストデータに対しては，僅かではあるが正解率が低下している．これに対し，学習データを混合した場合には，いずれのドメインのテストデータに対しても 100%近い正解率になっていることから，多様なドメインに対応した SBERT が構築できた可能性がある．

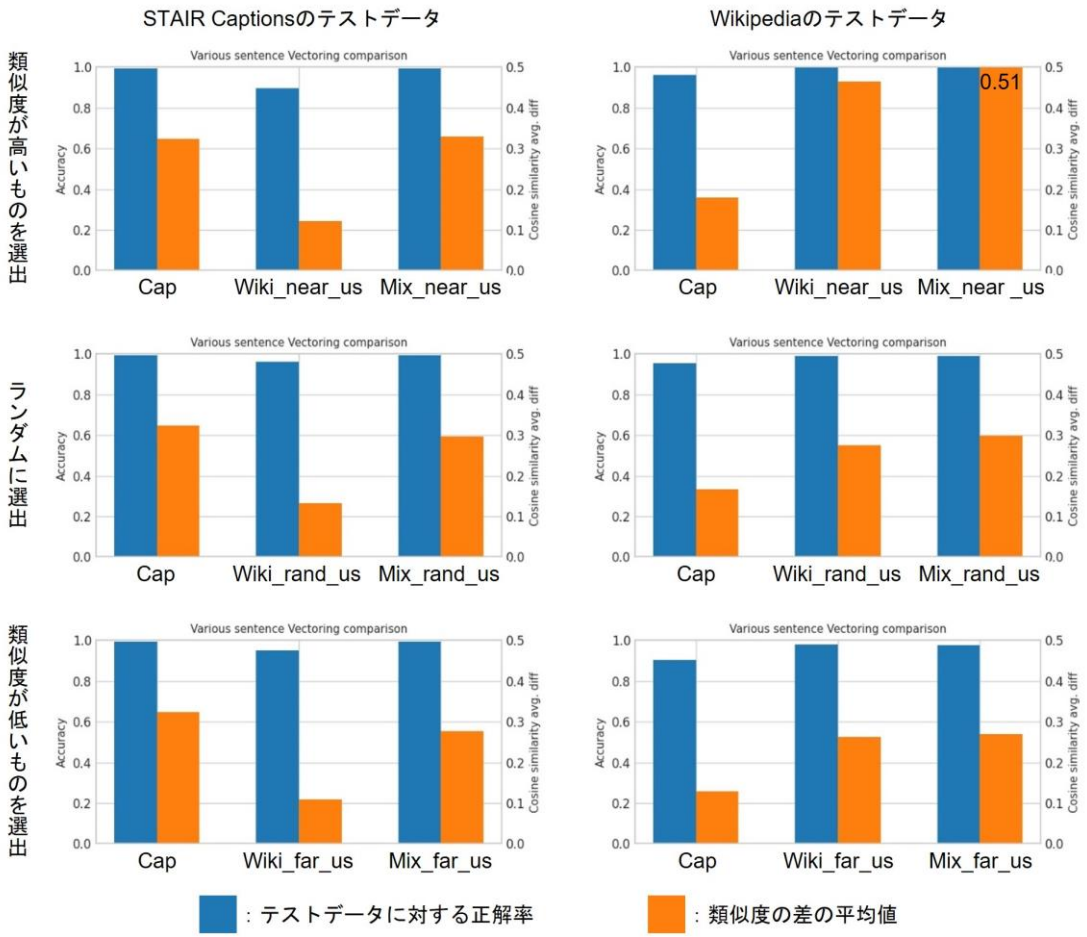

図 3　アンダーサンプリングありの学習データでファインチューニングした SBERT の性能比較

類似度が高いものをポジティブ文として選出した near と名の付くモデル群は，Wikipedia のテストデータにおいて類似度の差の平均値が他の SBERT には見られないほど高くなっている．その一方で，STAIR Captions のテストデータに対する正解率の低下が，rand や far と名の付くモデル群よりも大きいことが見受けられるため，類似度が高いものをポジティブ文として選出したことで，Wikipedia のドメインに対して過適合している可能性が考えられる．この STAIR Captions のテストデータに対する正解率の低下は，STAIR Captions の Triplet データと混合した Mix_near や Mix_near_us において改善している．しかしながら，Wikipedia テストデータにおいて類似度の差の平均値が非常に高くなっている傾向は，混合の有無やアンダーサンプリングの有無に関わらず生じており，依然として過適合している可能性が考えられる．

ポジティブ文を同じ見出しの文からランダムに選出した rand と名の付くモデル群や，あえて類似度が低いものをポジティブ文として選出した far と名の付くモデル群では，near と名の付くモデル群と比較して性能が劣化すると予想していたが，予想に反してそのような傾向は見られなかった．また，near と名の付くモデル群で見られたような Wikipedia のドメインに対して生じていると思われる過適合も見られない．したがって，Wikipedia のような比較的長い文でファインチューニングする場合において，ポジティブ文をどのように選定すべきか判断するためには，実際に未知のドメインの文に対する類似文検索の性能にどのように影響しているのか調査する必要がある．次節では，作成した SBERT を使用し，未知のドメインのデータベースに対する類似文検索を行い適切な応答ができるか評価する．

5. 未知のドメインでの質問応答テスト

SBERT の構築に使用したデータセットのドメインとは異なる未知のドメインのデータとしてある企業の社内規則を使用した．社内規則を使用することとしたのは，「～した場合は・・・」といった条件と帰結に分けられる文が多く含まれているためである．この社内規則から先行研究の手法[6]で条件部と帰結部を抽出した結果，推論データベースとして 598 のデータが得られた．

類似文検索の評価で使用する入力文と正解データとして設定した文を表 1 に示す．まず，「有給休暇を取得したい」という入力に対しては，「年次休暇等の有給休暇を取得する場合は」という文を発見することを想定している．この場合，提案システムは対応する帰結部である「本人の所定勤務時間に応じた賃金を支給する」という返答を行う．「有給休暇」という具体的な単語が含まれていることから，単語レベルの意味の類似性を捉えられていれば容易に発見できると考えられる．「パソコンを紛失した」も同様であり，「社員証、携帯電話、パソコン等を紛失した場合は」という文を発見することは容易であると思われる．これに対し，「社員証を無くした」という入力から同じ文を発見

するためには，紛失したものを示す単語が「パソコン」だけでなく，文の冒頭の「社員証」も含むことを文の意味として理解できていなければならない．また，「無くした」という表現が「紛失した」と類似していることも理解できている必要がある．このことから，類似文検索の難易度が高いと予想した．次に「出張する」については「出張」を含む文が多いため，「社員が出張を命ぜられた場合は」という最も適切と思われる文を発見できるか確認するために設定した．

表 1　類似文検索の評価で使用した入力文と正解データ

入力文	正解データ	
	条件部	帰結部
有給休暇を取得したい	年次休暇等の有給休暇を取得する場合は	本人の所定勤務時間に応じた賃金を支給する。
パソコンを紛失した	社員証、携帯電話、パソコン等を紛失した場合は	実費相当として以下の金額を本人から徴収する。
社員証を無くした		
出張する	社員が出張を命ぜられた場合は	別に定める「出張旅費規程」により旅費を支給する。

類似文検索では，構築した SBERT で入力文と条件部とをベクトル化し，その COS 類似度が高い上位 3 つを出力する．そのうち正解データの順位が 1 位だった場合は 3 点，2 位は 2 点，3 位は 3 点，上位 3 つに入っていなかったら 0 点と点数付けしてモデルを評価した．その結果を表 2 に示す．その結果，Mix_rand_us が全ての入力文において正解データとの COS 類似度が最も高く，最高得点であった．

表 2　類似文検索の評価結果

モデル名	入力文				合計
	有給休暇を取得したい	パソコンを紛失した	社員証を無くした	出張する	
Cap	3	3	0	0	6
Wiki_near	0	0	0	0	0
Wiki_rand	3	0	0	0	3
Wiki_far	0	0	0	0	0
Mix_near	0	0	0	0	0
Mix_rand	3	2	0	0	5
Mix_far	2	3	0	0	5
Wiki_near_us	0	0	0	0	0
Wiki_rand_us	3	3	0	0	6
Wiki_far_us	3	0	0	0	3
Mix_near_us	0	0	0	0	0
Mix_rand_us	3	3	3	3	12
Mix_far_us	3	2	2	0	7

次に，点数が最も高かった Mix_rand_us において，模範的な正解が特に存在しない「お金が欲しい」という文を入力し，その出力の上位 3 つがどのような文か確認した．その結果を表 3 に示す．

表 3　Mix_rand_us を使用した入力文：「お金が欲しい」の類似文検索結果

順位	条件部	帰結部
1	死亡一時金を受けることができる遺族がいないときは	死亡した者の個人別管理資産額に相当する金銭は死亡した者の相続財産とみなす。
2	社員が結婚したときは	別表に定める結婚祝金を贈る。
3	毎月 1 日時点で入居している場合に	その月分の借上社宅使用料を給与支給日に会社に支払わなければならない。

6. 考察

6.1 類似文検索のための SBERT の構築方法

まず，Caption の学習データのみでファインチューニングした場合（Cap），単語レベルでの意味の類似性を捉えられており，「有給休暇」や「パソコンを紛失」といった入力文に存在する単語やフレーズがそのまま含まれる条件部は正しく発見できていた．一方，Wikipedia のデータのみでファインチューニングしたモデル（Wiki と名の付くモデル群）では，ポジティブ文の選定方法に関わらず類似文検索の性能が Cap に劣っている．これは，Wikipedia の学習データのみではドメインレベルの類似性の学習に留まり，単語レベルの類似性が十分に獲得できていないことが考えられる．この性質は Wiki_near において顕著であり，3.5 節の性能比較で述べたように，Wikipedia の学習データに過適合した結果であると思われる．これに対し，Wiki_rand や Wiki_rand_us では性能が改善しており，同じ見出しからランダムにポジティブ文を選出することが過適合を防ぐ上で有効であると考えられる．

ランダムに選出する方法や，あえて類似性が低い文を選出する方法で作成した Wikipedia の学習データと，STAIR Captions の学習データとを混合した場合（Mix と名の付くモデル群）では類似文検索の性能が高いモデルが多い傾向がうかがえる．特に Mix_rand_us は全ての入力文に対して適切な類似文を発見できており，「社員証を紛失した」という入力文から「社員証、携帯電話、パソコン等を紛失した場合は」という条件部を唯一発見することができていた．これは他のモデルでは一切発見できておらず，Mix_rand_us は文の意味を捉える性能が非常に高いと言える．また，Mix_far_us においても，ほとんどの入力文に対して第 2 位以内に適切な条件部が発見できている．

このような性能の向上は Mix_near や Mix_near _us ではほとんど見られない．その原因として，類似性が高い文を選出する上で Cap を使用したことで，Wikipedia の学習データから獲得できる文の意味の類似性が，STAIR Captions の学習データから獲得できる文の意味の類似性と近いものとなり，混合することで得られるはずの多様性が失われた可能性が考えられる．さらに，前述の通り Wikipedia の学習データから獲得できるのはドメインレベルの意味の類似性である可能性があり，これに過適合することによって STAIR Captions の学習データから獲得できる意味の類似性の効果が薄まったことも考えられる．これに対し，ランダ

ムに選出する方法や，類似性が低いものを選出する方法で作成した学習データと混合することは，STAIR Captions の学習データのみでは獲得しにくい意味の類似性を獲得できるように学習データを改善することに繋がった可能性がある．しかしながら，あえて類似性が低いものを選出することは，ノイズとなるデータが増加したり，Wikipedia の学習データから獲得できる意味の類似性の範囲を限定することに繋がったり，類似文検索の性能に悪影響を与える場合もあると考えられる．以上の結果から，様々なドメインに適用できる多様性を持った類似文検索モデルを構築する上で，異なる性質を持つ学習データを混合すること，その際，一方の学習データに過適合しないように均等なデータ数で混合することが有効であると考えられる．

6.2 推論データベースに基づく質問応答の可能性

推論データベースの条件部に対する類似文検索で最も性能が高かった SBERT モデルである Mix_rand_us において社内規則のドメインに必ずしも合致していない「お金が欲しい」という文を入力した結果（表 3），第 1 位は「死亡一時金を受けることが…」というお金に関連する条件部を発見できていた．しかしながら，その帰結部を返答したとしても適切とは言えない．したがって，異なるドメインの複数の推論データベースの中から，いずれを検索対象とするかを入力文から適切に選択する必要がある．BERT による質問応答システムにおいては，検索対象とする Passage を入力文から選択する手法が提案されており，その手法を利用することが有効に働くかもしれない．

類似文検索結果として第 2 位，第 3 位に出力された条件部は一見お金に関係が無いように見える．しかしながら，その帰結部は「別表に定める結婚祝金を贈る」「その月分の借上社宅使用料を給与支給日に会社に支払わなければならない」というお金に関係したものであった．類似文検索の対象は条件部のみであったが，「結婚」や「入居」することがお金と関連性が高いということも含めて文の意味の類似性を学習できていた可能性がある．このことから，入力文が 5W1H のような明確な質問文でなくても，文の意味の類似性を捉えて疑似的な演繹推論を行うことによって，ユーザのふとしたつぶやきから，意図を汲み取ったような応答ができる音声アシスタントを構築できるかもしれない．

本研究では，推論データベースの条件部のみを検索対象としたが，帰結部を検索対象とした場合，より返答の多様性を高められる可能性がある．例えば，ユーザから「パスポートを取ったんだ」という発話が入力されたとする．そして，推論データベースに「海外旅行に行く場合は」という条件部と，「パスポートが必要です」という帰結部が含まれていれば，類似文として「パスポートが必要です」という帰結部を発見し，それに対応する条件部を元に「海外旅行に行くんですか？」という仮説推論に基づく応答ができるかもしれない．仮説推論に基づく応答を条件部から文末表現の変換等で生成できるか検証することは今後の課題である．

7. 制限事項

本研究では，BERT のファインチューニングに使用するデータセットを，STAIR Captions と Wikipedia データベースから作成した．それらのデータセットだけでなく，その他のドメインのデータセットとさらに混合することで，類似文検索の精度が高まるか，対象となるドメインの多様性を広げられるか検証することは今後の課題である．

Open Domain な質問応答システムを構築する上で，多数の推論データベースの中から検索対象とするものを入力文からいかに選択するかは本研究では検証できていない．先行研究の手法[4][5]を試すことや，新たな手法を開発することで，様々な入力文に適切に応答できるかテストすることは今後の課題である．

推論データベースは既存の文章から条件部と帰結部を抽出して作成するため，帰結部をそのまま返答文として使用すると，音声アシスタントのパーソナリティに合致しない場合がほとんどであると思われる．これは既存の質問応答システムを音声アシスタントに実装する上で共通の問題である．音声アシスタントのパーソナリティに合わせて返答文の表現を変換する方法を検討し，有効性を検証することは今後の課題である．

8. まとめ

本研究で提案した質問応答システムは，知識となる文を条件部と帰結部に分けて保存した推論データベースから，ユーザの入力文に類似する条件部を発見し，帰結部をシステムからの応答文とすることで，演繹推論に基づいた返答を疑似的に行うものである．その利点は，ユーザからの入力が何気ないつぶやきなど明確な質問文でなくても応答できる点であり，関連する知識をシステムから能動的にユーザに与える発話を行える可能性がある．本研究では，この提案システムを構築する上で重要となる類似文検索モデルをいかに構築するか実験的に検証した．

知識を条件部と帰結部に分けて持つ推論データベースはドメインによって必ずしも大量に取得できるわけではない．本研究で使用した社内規則というドメインは条件部と帰結部に分けられる文を比較的多く含んでいるが，それであっても高々約 600 文程度であり，言語系の機械学習モデルの学習には不十分である．そこで，大量に入手可能な学習データを混合することで，未知のドメインの類似文検索の精度を高められるか検証した．日本語学習済み BERT を様々な条件でファインチューニングして SBERT を構築し，類似文検索の性能を比較評価する実験を行った結果，Wikipedia の記事から作成したデータセットと，同一の画像につけられたキャプションから作成したデータセットを，データ数を揃えて（アンダーサンプリングして）混合することで，学習データに含まれていないドメインである社内規則の推論データベースにおいても類似文検索の性能を高めることができた．最も類似文検索の性能が高い SBERT は，入力文の意味を単語レベルだけでなく文レベルで捉えられており，意味が近い条件部を発見することで，対応する帰結部を用いて適切な応答が行える可能性を示した．

謝辞

本研究は，JSPS 科研費 JP19K12081，JP20H05564，JP21J15349 の支援を受けた．

参考文献

[1] Stefan Kojouharov, Ultimate Guide to Leveraging NLP & Machine Learning for your Chatbot, https://chatbotslife.com/ultimate-guide-

to-leveraging-nlp-machine-learning-for-you-chatbot-531ff2dd870c (参照 2022-6-17).
[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, arXiv:1810.04805, (2019). https://arxiv.org/abs/1810.04805v2
[3] Shiyue Zhang, Mohit Bansal, Addressing Semantic Drift in Question Generation for Semi-Supervised Question Answering, arXiv:1909.06356, (2019). https://arxiv.org/abs/1909.06356
[4] Wei Yang, Yuqing Xie, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, Jimmy Lin, End-to-End Open-Domain Question Answering with BERTserini, arXiv:1902.01718. (2019). http://dx.doi.org/10.18653/v1/N19-4013
[5] Zhiguo Wang, Patrick Ng, Xiaofei Ma, Ramesh Nallapati, Bing Xiang, Multi-passage BERT: A Globally Normalized BERT Model for Open-domain Question Answering, arXiv:1908.08167. (2019). https://arxiv.org/abs/1908.08167v2
[6] Truong-Son Nguyen, Le-Minh Nguyen, Satoshi Tojo, Ken Satoh, Akira Shimazu, Recurrent neural network-based models for recognizing requisite and effectuation parts in legal texts, Artificial Intelligence and Law, Vol.26, pp.169–199, (2018). https://doi.org/10.1007/s10506-018-9225-1
[7] Nils Reimers, Iryna Gurevych, Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks, arXiv:1908.10084, (2019). https://arxiv.org/abs/1908.10084v1
[8] J.S. McCarley, Rishav Chakravarti, Avirup Sil, Structured Pruning of a BERT-based Question Answering Model, arXiv:1910.06360, (2019). https://arxiv.org/abs/1910.06360v3
[9] Chen Qu, Liu Yang, Minghui Qiu, W. Bruce Croft, Yongfeng Zhang, Mohit Iyyer, BERT with History Answer Embedding for Conversational Question Answering, arXiv:1905.05412. (2019). https://arxiv.org/abs/1905.05412v2
[10] Akari Asai, Kazuma Hashimoto, Hannaneh Hajishirzi, Richard Socher, Caiming Xiong, Learning to Retrieve Reasoning Paths over Wikipedia Graph for Question Answering, arXiv:1911.10470. (2019). https://arxiv.org/abs/1911.10470v2
[11] 橋口 友哉, 山本 岳洋, 藤田 澄男, 大島 裕明, CQA コンテンツからの状況が類似する悩みの検索, 人工知能学会論文誌, Vol.36, No.1, p.WI2-B_1-13, (2021). https://doi.org/10.1527/tjsai.36-1_WI2-B
[12] Wataru Sakata, Tomohide Shibata, Ribeka Tanaka, Sadao Kurohashi, FAQ Retrieval using Query-Question Similarity and BERT-Based Query-Answer Relevance, arXiv:1905.02851, (2019). https://arxiv.org/abs/1905.02851v2
[13] Tong Guo, Huilin Gao. Revisiting Semantic Representation and Tree Search for Similar Question Retrieval, arXiv:1908.08326, (2019). https://arxiv.org/abs/1908.08326v8
[14] STAIR Captions. http://captions.stair.center/ (参照 2022-6-17).
[15] 鵜野 和也, はじめての自然言語処理（第 9 回 Sentence BERT による類似文章検索の検証）, (2020). https://www.ogis-ri.co.jp/otc/hiroba/technical/similar-document-search/part9.html (参照 2022-6-17).
[16] NICT BERT 日本語 Pre-trained モデル. https://alaginrc.nict.go.jp/nict-bert/index.html (参照 2022-6-17).