質問応答におけるユーザ入力表現のドメイン固有の表現へのマッピング

TSENG YI KAI[1]　徳永健伸[1]　横野光[2]
[1] 東京工業大学 [2] 明星大学
tseng.y.ab@m.titech.ac.jp　take@c.titech.ac.jp　hikaru.yokono@meisei-u.ac.jp

概要

あるドメインの初心者はそのドメイン固有の用語を知らないことが多く，別の単語で言い換えることがあり，質問応答システムなどを利用する際の問題になる．この問題に対して，本論文では BERT を用いて単語の埋め込みを計算し，コサイン類似度により初心者が入力した表現とドメイン固有の表現を対応づける手法を提案する．Minecraft を対象として提案手法の評価を行った．

1　はじめに

あるドメインにおいてユーザが問題に遭遇したとき，キーワードを検索エンジンに入力して膨大な情報から欲しい情報を自分で探すより，自然言語で質問して回答を得る方が簡単である．しかし，質問者がそのドメインの初心者である場合，ドメイン固有の用語を知らないために，それを他の単語で表現することがある．そのような表現を含む質問に対してシステムは適切な回答を返せないことが考えられる．

そこで本論文では，あるドメインの質問応答システムにおいて初心者が使用することを想定し，質問中の表現をドメイン固有の正式な名称に対応づけるシステムの構築を行う．具体的なドメインとして有名なコンピュータゲームである Minecraft[1)] を対象とする．

Minecraft には多様なオブジェクトが存在し，プレイヤーが取れる行動が多様で自由度が高い．そのため，何をすれば良いか分からないという初心者ユーザが多く，コミュニティサイトでは様々な質問のやり取りがなされている．

Minecraft における質問応答システムに関して，Dumont et al. [2016] はコーパスを作成し，質問応答システムの実現可能性を示した．しかし，Minecraft には特有の用語が存在するため，前述の問題が起こりうる．例えば，“melon” という物体はスイカのような見た目であることから “watermelon” と，また，スイッチの機能を持つ “lever” を “switch” と言い換えられることがある．既存の質問応答システムはこのように使用されている表現に対応できない可能性がある．そこで，例えば “How do I craft a switch?” という質問文の “switch” という単語を正式な名称である “lever” に対応づけることで，ドメインの知識をあまり持っていない初心者でも使える質問応答システムの実現を目指す．

このような対応付けを実現するために，単語の埋め込みを計算し，コサイン類似度により対応する用語を判断する．埋め込みには，Word2Vec [Mikolov et al., 2013]，Global vectors [Pennington et al., 2014]，BERT [Devlin et al., 2018] といった数多くの手法が使われているが，本論文では文脈を考慮できる BERT を用いて，初心者による表現をより精度よく正式な名称に対応づける．

2　関連研究

本論文で扱う現象は言い換えの一種とみることができる．Niu et al. [2021] は，質問応答システムの精度向上を含む幅広い応用を目指し，教師なし学習によるパラフレーズの生成の手法を提案した．教師なし学習は手動によるラベル付きの必要がないため，より簡単に構築できる．Behrendt and Harmeling [2021] は，BERT を拡張し，ArgueBERT という二つの文の類似度を計算することによりパラフレーズを検出するモデルを提案した．Wang et al. [2021] は，ドメイン特有の知識も考慮した上で BERT を用いてパラフレーズを検出する方法を提案した．

これらに対し，提案システムでは事前学習済み BERT の埋め込みを利用し，コサイン類似度により

1）https://www.minecraft.net/

関連の用語を出力する．Niu らは教師なし学習を利用するのに対し，本研究では事前学習済み BERT を用いるため，手動によるラベル付けのみならず Finetuning も必要なく，より簡単に構築できる．

また，Behrendt らと Wang らは文単位でパラフレーズか否か判定するのに対し，提案システムではランキングをもとに質問文の表現に対応するドメイン固有の表現を提示する．これは，初心者による言い換えは必ずしも正確な言い換えとは限らないため，類似している用語をいくつか提示することにより，柔軟に対応できるからである．

3 提案システム

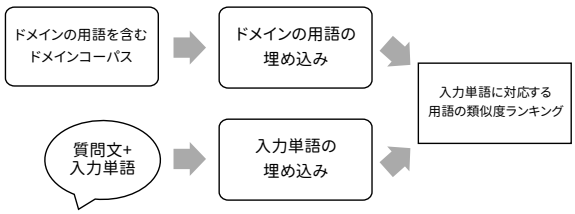

図 1 提案システムの概要

提案システムは，図 1 に示すようにドメイン固有の表現 (用語) に対して事前にドメインの知識を持つユーザが作成したテキスト (ドメインコーパス) を用いて，その埋め込みを計算しておく．そして，入力された質問文中の単語に対して，その埋め込みを計算し，ドメイン固有の表現とのコサイン類似度を求め，ランキングの 1 位のものを対応として出力，または，上位のランキング結果を提示する．テキストの分割には BERT の tokenizer を用いる．

3.1 埋め込み

埋め込みの計算は事前学習済み BERT (bert-base-uncased)[2] を用いる．単語の埋め込みは BERT が出力する 768 次元の最後の隠れ層を使う．

質問文中の表現の埋め込みと用語の埋め込みのコサイン類似度の計算方法は次の 2 種類を用いる．

- Mean-of-Posts：各用語 w に対し，ドメインコーパスに出現した w の埋め込みの算術平均を w の埋め込みとし，それと入力単語の埋め込みのコサイン類似度を求める
- Max-Post：各用語 w に対し，ドメインコーパスに出現した w のそれぞれの埋め込みと入力の単語の埋め込みとのコサイン類似度を計算し，その最大値を w とのコサイン類似度とする

2） https://huggingface.co/bert-base-uncased

3.2 Tokenization

文を BERT に入力する際に，tokenizer を用いて文をトークンに分割する必要があるが，用語によって tokenizer の動作が異なることがある．

複数形の扱い 単語の複数形に対して 2 種類の分割が存在する．一部の単語の複数形は単数形と語尾の-s と 2 個のトークンに分割される．例えば，“lever” の複数形 “levers” は “lever” と “##s” に分割される．この場合，単数形の埋め込みがそのまま使えるため，“##s” を無視して単数形の埋め込みを考慮すればよい．一方，複数形がそのまま 1 個のトークンになる単語も存在する．例えば，“block” の複数形 “blocks” は “block” と “##s” と 2 個のトークンではなく，そのまま “blocks” という 1 個のトークンになる．この場合，複数形のトークンは単数形のトークンと異なるため，求めた埋め込みに差が出てしまい，単数形の埋め込みのみ考慮すると複数形に対応できない恐れがある．そこで，用語の複数形にも対応できるように，複数形が単数形と異なるトークンに分割される場合，単数形だけでなく，複数形の埋め込みも求め，用語 w との類似度として，w の単数形との類似度と w の複数形との類似度のうち高い方と選ぶ．

マルチトークン 単語が複数のトークンに分割されたり，複数の単語からなる用語が存在するため，1 つの表現が複数のトークンで構成されることがある．例えば，“ender pearl” は “end”，“##er”，“pearl” と 3 個のトークンに分割される．そのため，類似度の計算においてはマルチトークンの表現への対策が必要となる．マルチトークンの表現の埋め込みには次の 2 種類の計算方法を用いる．

- Last-Token：最後のトークンの埋め込みを表現全体の埋め込みとする．
- Mean-of-Tokens：全てのトークンの埋め込みの算術平均を全体の埋め込みとする．

4 評価

4.1 データセット

Minecraft を対象のドメインとして提案システムの評価を行う．考慮する Minecraft の用語としては Dumont et al. [2016] が提案した質問応答コーパスで列挙された 443 個のエンティティを採用する．いく

つかのエンティティには別称が付いている[3]ため，これらの別称を含めて合計 465 個の用語を対象とする．

ドメインコーパスとして Minecraft に関する Reddit の投稿[4]を用い，465 個の用語に対する埋め込みを計算した．Reddit からは 76437 投稿が取得でき，その中で 465 語のうち 457 語，443 エンティティのうち 438 エンティティが 1 回以上出現している．BERT による Minecraft の用語 w の埋め込みを求める際には w を含む文を入力する必要があるため，Reddit の投稿から w が出現した投稿のタイトルを利用した．また，評価用のデータとして Dumont らも使用した GameFAQs[5]という QA サイトの Minecraft に関する質問文のタイトルを用いた．このサイトからは 986 個の質問が取得できた．

4.2 結果と考察

評価データの各表現に対し，457 個の Minecraft の用語とのコサイン類似度ランキングを作成した．埋め込みが有効であれば，評価データの質問文に含まれる Minecraft の用語に対する類似度ランキングにおいて，その用語自体が 1 位になるはずである．例えば，“How to get glass?” という質問文の “glass” という表現に対して用語の類似度ランキングを作成すると，“glass” 自体が Minecraft の用語であるため，“glass” がランキングの 1 位になることが期待される．これを利用し，埋め込みの有効性を検証するために，GameFAQs の質問文で合計 598 回出現した 126 種類の用語自体の順位を分析した．

表現間の類似度の計算は 3.1 で言及した Mean-of-Posts と Max-Post の 2 種類を用い，マルチトークンの表現の埋め込みは 3.2 で言及した Last-Token と Mean-of-Tokens の 2 種類を用い，これらの組み合わせである 4 種類の手法に対して評価を行った．

各手法において用語自体が 1 位になった割合を表 1 に，用語の順位の分布と 1 位とならなかった用語の類似度の分布を図 2 に示す．

Max-Post & Mean-of-Tokens の精度が最も良いが，Max-Post は用語の個々の埋め込みとの類似度を計算するためストレージや計算能力が必要となる．この点を考慮すると，Mean-of-Posts & Last-Token の精度が良いと見ることができる．

3） 例えば “thunderstorm” というエンティティは “lightning” という別称を持つ．
4） https://www.reddit.com/r/Minecraft/
5） https://gamefaqs.gamespot.com/pc/606524-minecraft

表 1 1 位になった用語の割合

手法	1 位の割合
Mean-of-Posts & Last-Token	577/598 (96.5%)
Max-Post & Last-Token	559/598 (93.5%)
Mean-of-Posts & Mean-of-Tokens	574/598 (96.0%)
Max-Post & Mean-of-Tokens	588/598 (98.3%)

また，BERT は文脈を考慮した埋め込みを出力するため，入力の文に誤字が含まれると，得られる埋め込みが変化し，特に Max-Post を利用する場合ではランキングが大きく変化してしまう．一方，Mean-of-Posts を利用する場合，投稿間の埋め込みの平均を取るため，安定した埋め込みが得られると考えられる．

入力の単語と用語が共通語尾を持つ場合，埋め込みが近くなり，順位上で完全に分離できず，1 位にならないことがある．しかし，1 位ではなくても比較的高い順位に用語が表れていれば，上位の用語を提示することでユーザに適切な用語の発見を促すことができると考えられる．

Mean-of-Posts & Last-Token

最下位となった用語の順位は 8 位であった．用語自体が 1 位にならなかった事例では，入力の単語と共通の語尾を持つ用語が上位になった．8 位になった事例は “Do monsters still attack in creative mode?” であり，入力の表現は “creative mode” だった．用語との類似度は 0.8186 であり，このときの 1 位は “adventure mode”(類似度 0.8807) だった．2〜7 位も同様に “mode” という語尾を持つ用語であった．

1 位にならなかった用語の類似度は全て 0.7 以上と高いため，質問応答システムは 1 位の単語のみならず，全て類似度の高い用語を関連の用語としてユーザに提示することができる．

Max-Post & Last-Token

最下位となった用語の順位は 137 位であった．その事例は “Blockes will no drop blockes after I break them?” という質問文であり，入力の表現は “Block” だった．“Block” という用語自体との類似度は 0.6074 だったが，1 位が “Spider”(類似度 0.7847)，2 位が “Mob”(類似度 0.7737) であった．この事例の “Blockes” という単語は，“Block” の複数形と思われるが，正しい綴りは “Blocks” である．この誤字により埋め込みが大きく影響されてしまい，正解が

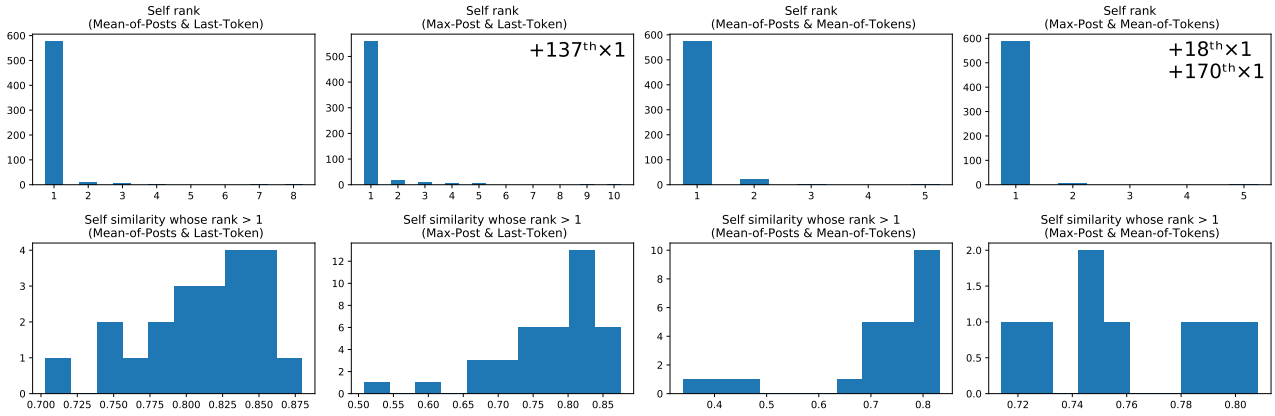

図 2　順位の分布 (上図) と 1 位とならなかった用語の類似度の分布 (下図)

得られなかったと考えられる．実際に “Blockes” を “Blocks” に修正すると，“Spider” との類似度が 0.3052 に，“Mob” との類似度が 0.3583 に下がり，“Block” 自体との類似度が 0.6288 に上がった．このことから，この手法は誤字の影響を受けやすいと言える．

Mean-of-Posts & Mean-of-Tokens

最下位となった用語の順位は 5 位であり，Mean-of-Posts & Last-Token と比べると，全体的には良い性能となり，最下位の用語の順位は良くなったが，1 位でない単語の類似度は低くなった．入力が “Mob spawners mayhem” という質問文で入力の単語が “Mob” である事例では，Mean-of-Posts & Last-Token において 1 位が “Mob”(類似度 0.3393) であったが，Mean-of-Posts & Mean-of-Tokens において “Mob” が 2 位となり，1 位がマルチトークンである lava slime(類似度 0.3862) となった．これは，トークン間の平均を取ったことにより，埋め込みに食い違いが発生したと考えられる．

Max-Post & Mean-of-Tokens

最下位となった用語の順位は 170 位だった．順位が極端に低い単語は 170 位と 18 位の 2 個であり，両方とも入力に誤字が含まれていた．170 位は Max-Post & Last-Token の項で言及した “Blockes” の例であり，18 位は “Wheather”(“Whether” の誤字) であった．これらを除外すると，最下位の用語の順位は 5 位になり，1 位でない単語の類似度は全て 0.7 以上と高くなるため，質問文に誤字が含まれていなければ，Max-Post & Mean-of-Tokens による埋め込みは有効であると考えられる．

4.3　異なる表現間のマッピング

4.2 では入力表現に対して，対応する Minecraft の用語が 1 位となることを正解として評価したが，実際に対話システムで使うことを考えると 2 位以下の候補を利用する可能性もある．そこで，4.2 で対象とした表現を除いた合計 1,310 個 (543 種類) の名詞を対象としてランキングを作成し，1 位の用語との類似度が高い名詞を分析した．

類似度が 0.8 以上になった名詞を確認したところ，多くは “game” → “the game”，“mode” → “hardcore mode” といった，用語と共通語尾を持つ名詞であり，類似度が 0.7 程度の単語を確認したところ，“circuits” → “redstone wire”，“farm” → “plant” といった，関連用語の対応が得られた．また，数は少なかったが，“space” → “inventory slot” といった我々が想定しているような言い換えも得られた．

5　おわりに

本論文では，特定ドメインにおける初心者が用いる表現をドメイン固有の表現に対応づけるシステムを提案した．具体的なドメインとして Minecraft を対象とし，コミュニティサイトのデータを用いて評価を行い，埋め込みを計算する 4 種類の手法を比較し，その特性や有効性を考察した．

実際の対話システムの環境で，より多くの事例を収集するために Ogawa et al. [2020] の基盤を用いて初心者と経験者のペアを組ませて対話を収集し，事例の分析やシステムの改良を行う予定である．

謝辞

本研究は JSPS 科研費 JP19H04167 の助成を受けた.

参考文献

M. Behrendt and S. Harmeling. Arguebert: How to improve bert embeddings for measuring the similarity of arguments. In **Proceedings of the 17th Conference on Natural Language Processing (KONVENS 2021)**, pages 28–36, 2021.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. **arXiv preprint arXiv:1810.04805**, 2018.

C. Dumont, R. Tian, and K. Inui. Question-answering with logic specific to video games. In **Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)**, pages 4637–4643, 2016.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. **arXiv preprint arXiv:1301.3781**, 2013.

T. Niu, S. Yavuz, Y. Zhou, N. S. Keskar, H. Wang, and C. Xiong. Unsupervised paraphrasing with pretrained language models. In **Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing**, pages 5136–5150, 2021.

H. Ogawa, H. Nishikawa, T. Tokunaga, and H. Yokono. Gamification platform for collecting task-oriented dialogue data. In **Proceedings of the 12th Language Resources and Evaluation Conference**, pages 7084–7093, 2020.

J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In **Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)**, pages 1532–1543, 2014.

H. Wang, F. Ma, Y. Wang, and J. Gao. Knowledge-guided paraphrase identification. In **Findings of the Association for Computational Linguistics: EMNLP 2021**, pages 843–853, 2021.