長期間の過去文脈を効果的に活用した雑談対話システム

Open-Domain Dialogue System Utilizing Long-Term Past Contexts

高﨑環 [1*]　吉永直樹 [2]　豊田正史 [2]
Meguru Takasaki[1]　Naoki Yoshinaga[2]　Masashi Toyoda[2]

[1] 東京大学大学院情報理工学系研究科
[1] Graduate School of Information Science and Technology, the University of Tokyo
[2] 東京大学生産技術研究所
[2] Institute of Industrial Science, the University of Tokyo

Abstract: This study proposes a task-specific retriever for effectively extracting core fragments of dialogue histories that are useful to reply to a given dialogue context for open-domain dialogue systems. We experimentally confirm the advantage of our retriever against the existing session-based retriever on a GPT-2-based dialogue system trained with a large-scale Twitter dataset.

1　はじめに

スマートフォンやスマートスピーカーを介し，知的対話アシスタント（例: Apple Siri, Amazon Alexa）と日常的に会話を行う機会が増えている．これらの会話においてユーザのエンゲージメントを高めるためには雑談をサポートすることが有効である [3]．雑談で扱う多様な話題に対応するには，SNS 上の大規模対話ログを使用してモデルを学習することが有効であるが [16]，深層学習を学習に用いてもなお，応答品質には課題が残る．その理由として，人間同士の会話が過去の会話を踏まえて行われるのに対し，対話システムでは多くの場合，直前の会話のみを参照し応答を生成するため，応答の自由度が高過ぎる [18] ことが挙げられる．

この課題に対し，対話システムと特定ユーザの間の継続的対話を想定して，過去の対話セッションを参照しながら応答を生成する研究が行われている [22, 23, 2]．しかしながら，深層学習に基づく応答生成モデルでは入力長の制限が厳しく [1, 17]，現在の対話文脈に類似したセッションを参照する手法では応答性能の改善は限定的である [22]．そのため，過去の対話セッションに含まれる情報を，要約 [22] やペルソナ文集合 [23, 2] などの形で圧縮して入力する手法が試みられているが，これらの手法では，圧縮した情報を教師として与える必要があり，学習データの大規模化が難しい．

そこで本論文では，情報圧縮用の追加の学習データを必要とせず，長期間の継続的な対話履歴の情報を効果的に参照する雑談対話システムを提案する（図 1）．我々が提案する手法では，現在進行中の対話文脈を query とし，細分割した過去履歴を，key と履歴に紐づく query との類似度を基準として検索し，応答生成に有益な過去履歴を抽出する．query と key は，同セッションに登場する対話文脈同士の距離を最小化するように学習した Sentence-BERT [15] を用いてベクトル化し，query/key の効果的な単位を検証する．

実験では，継続的に収集した大規模 Twitter データから，特定ユーザ同士が 10 セッション以上に亘り会話するデータセットを構築し，提案手法の有効性を検証した．自動評価と人手評価の結果，GPT-2 [14][6]ベースの雑談対話システムにおいて，提案する抽出器が既存のセッション単位での抽出器を上回ることを確認した．この結果から，過去の対話文脈を適切な粒度で用いることで，応答性能が向上することが考えられる．

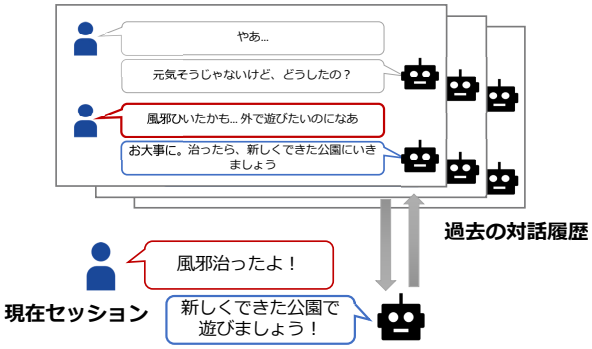

図 1: 過去の対話セッションを用いた応答生成．過去セッションには，応答生成に役立つ現在の対話文脈と関連した発話（赤枠）やその発話に対する応答（青枠）も含まれるが，無関係の発話（灰色枠）も多い．

*連絡先：東京大学生産技術研究所
〒 153-8505 東京都目黒区駒場 4 丁目 6 - 1
E-mail: takasa-m@tkl.iis.u-tokyo.ac.jp

2 関連研究

本節では，まず，検索で取得した対話ログを用いて応答を生成する手法を紹介する．次に，本研究と同様に，長期間の対話文脈を考慮した応答生成手法を説明する．最後に，他の言語処理タスクにおいて，タスクを解くのに役に立つ知識を，学習データや生テキストから検索して参照する手法を紹介する．

2.1 検索に基づく応答生成

Pandey ら [12] は，特定ドメインの会話で見られる典型的なやり取りに注目し，対話ログから現在の対話文脈と類似する対話履歴を検索して後続の応答を参照しつつ，文脈に即した応答を生成する手法を提案した．Wu ら [21] は，雑談において多様な応答生成を行うために，検索で得られた対話文脈と現在の対話文脈の違いを考慮しながら，得られた対話文脈に後続する応答を編集して応答する手法を提案した．また Cai ら [4] は，同様の目的で，検索で得られた応答の内容を生成する応答に強く反映させるための工夫を提案している．

これらの研究が，類似する対話文脈に後続する応答を雛形として応答の生成に用いるのに対し，本研究は特定話者との今までのやり取りと矛盾なく応答を生成するために，過去の対話文脈を参照する．提案手法は，過去の対話文脈を埋め込むための追加のモジュールを必要とせず，事前学習済み生成モデルを活用することが可能である．

2.2 長期間の文脈を考慮した応答生成

Xu ら [22] は，長期間の継続的対話を想定した対話データセット (Multi-Session Chat (MSC)) を構築し，さらに過去の対話文脈から生成した要約を参照しつつ，応答を生成する手法を提案した．Xu ら [23] は，ユーザとチャットボット双方のプロフィール文を保持・参照・更新を行い，応答を生成する手法を提案した．これらの研究をもとに Bae ら [2] は，対話要約を動的に更新しつつ，要約を用いて応答を生成する手法を提案した．

これらの手法が話者の情報を圧縮するために追加の教師情報（要約，ペルソナ文）を必要とするのに対し，本研究で提案する対話文脈抽出器は対話ログのみから学習できる点に違いがある．

2.3 知識の検索に基づく自然言語処理モデル

オープンドメイン QA では知識源となる文書を検索する手法 [6, 9, 7] が広く用いられるようになっており，これらの手法の成功を受けて，他の言語処理タスクでもタスクを解くのに有用な知識を検索する手法が研究されている．Shinzato ら [19] は属性抽出を行う際に，入力に関係する属性の集合を学習データから検索して参照する手法を提案している．Wang ら [20] は，文章要約や質疑応答など様々なタスクに対し，訓練データから抽出した知識を参照することで，タスクの性能を改善した．Nishida ら [11] は，固有表現抽出において，モデルのタグ付けの確信度が低いエンティティのタグ付けを，それらのエンティティを query に用いて入手したテキストを参照し，修正する手法を提案している．

これらの研究から，検索に基づく言語処理モデルでは，情報を抽出する際の query/key/value をタスクに合わせて設計することが重要であると分かる．本研究では，過去の対話文脈から応答の生成に役立つ情報を得るために，key/value の設計方法を模索する．

3 Twitter 長期間対話データセット

長期間対話を想定して Xu ら [22] が構築した MSC データセットは，与えられた少数のプロフィール文を踏まえて会話を行う PersonaChat データセット [24] を拡張したものであるため，表面的な内容の会話が多い．また，データセットにおける過去の対話セッションの数は最大でも 3 セッションであり，応答生成モデルの最大入力長を大きく超過することはない．

そこで本研究では，より実際的なデータセットして，Twitter 上のユーザ間の会話を基に，長期間の雑談対話データセットを構築した．具体的には，Twitter API[1] を利用して，2011 年 3 月から研究室で継続的に収集したツイート[2]を元に，複数セッションからなる雑談対話データセットを構築した．本データセットは，Twitter API を用いて取得可能な形での公開を検討している．

実験では，データセット内のユーザ間の会話を人と対話システムの会話と見做し，対話システムは片方の話者の応答を生成するように学習，評価を行う．収集の際には，特定の 2 話者が交互に話すリプライツリーを，1 つの対話セッションとみなす（以下の説明では，特定の 2 話者間の対話セッション群を「エピソード」と表記する）．評価データを構築後，評価データと話者の重複がないように開発データを構築し，さらに評価・開発データ双方と話者の重複がないように訓練データを構築した．また，bot による自動投稿や URL，画像データはデータセットから除いた．極端に短いもしくは長い対話を除外するために，本データセットでは 5-30 ターンで構成される対話セッションを使用し，最終的

[1] https://developer.twitter.com/en/docs/twitter-api

[2] 26 人の著名な日本人ユーザを起点に，メンションやリツイートされたユーザを収集対象ユーザに加え，そのユーザのタイムライン（最新のツイート群）を user_timeline API を利用して継続的に収集したものである．

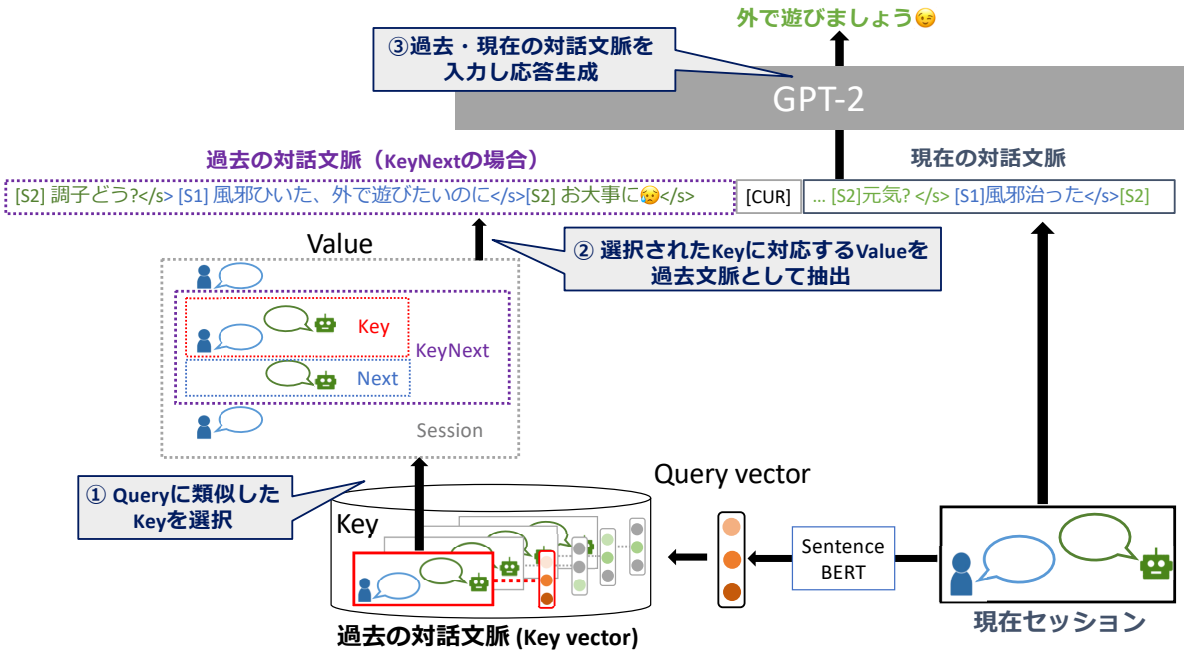

図 2: 提案手法の概観．現在の対話文脈から query を作成し，細分化された過去の対話文脈に紐づく key との照合（類似度計算）を行う．選択された key に対応する value を現在の対話文脈とモデルに入力し，応答を生成する．

	Train	Dev.	Test
全セッション			
収集期間	2011-2017	2018	2019
エピソード数	60,000	1,778	2,682
予測応答数	150,747	4,666	7,113
エピソードあたりの平均トークン数	2,593.65	2,654.47	2,561.63
エピソードあたりの平均セッション数	16.92	16.41	16.49
（1024 トークンを上回る割合）(%)	98.99	99.44	98.66
最終セッション（評価対象）			
セッションあたりの平均ターン数	6.65	6.89	6.89
セッションあたりの平均トークン数	146.37	151.51	146.66
過去セッション			
セッションあたりの平均ターン数	6.86	7.17	7.04
セッションあたりの平均トークン数	153.68	162.41	155.93

表 1: 使用した Twitter データセットの詳細

に 11-25 セッションを含むエピソードのみを収録した．データセットの統計量を表 1 に示す．

4 提案手法

本節では，長期間に亘り継続的に行われる雑談対話を想定して，過去の対話文脈を効果的に参照する応答生成モデルを提案する（図 2）．提案手法では，現在の対話文脈を query として過去の対話文脈を検索して抽出し，応答生成モデルの入力に追加する．本手法は，RAG [9] や FiD [7] とは異なり，過去の対話文脈を埋め込むための追加のエンコーダを必要としないため，GPT-2 [25] などのデコーダのみの応答生成モデルも活用できる．

4.1 過去文脈抽出器

提案する過去文脈抽出器は，与えられた現在の対話文脈を基に，Sentence-BERT を用いて過去セッションから有用な文脈を抽出する．まず，類似した対話文脈を抽出するための query/key/value の設計詳細を説明する (4.1.1 章)．その後，対話文脈を埋め込むための Sentence-BERT について説明する (4.1.2 章)．

4.1.1 過去の対話文脈の抽出

応答の生成に役立つ対話文脈 (value) を過去の対話セッションから検索・抽出するために，現在の対話文脈から query，過去の対話文脈から key を構築する．その後，query と key を Sentence-BERT でベクトル化し，query とのコサイン類似度が高い key を選択し，key に紐づけられた同セッション内の対話文脈を value として応答生成モデルに入力する．先行研究 [22] では，query/key はいずれもセッション単位であり，key を value としても用いているが，1 つの対話セッション内でも会話の話題は大きく移り変わることがある．そのため，セッションを value として入力すると応答生成に役に立たない対話文脈が含まれることとなり，入力長を無駄に圧迫する可能性がある．そこで本手法では，value としてセッションより細かい単位を用いた場合の性能を検証する．具体的に，実験では後述するように query/key を 1-3 発話で構成し，value として以下の 4 種類の単位（図 2）を比較する．

Session key を含むセッション全体を過去の対話文脈として入力する [22]．現在の対話文脈と無関係な情報が多く含まれやすい点に注意されたい．

Key key をそのまま value として入力する．query に類似した key の情報は，現在の対話文脈の理解に有用であると考えられる．

Next key に後続する一発話を入力する．現在の対話文脈に対する応答の雛形として参考になると考えられる．

KeyNext key と後続する発話を連結して入力する．query と key の類似性を踏まえて key に後続する発話を活用できると期待される．

4.1.2 Sentence-BERT を用いた対話文脈埋め込み

現在と過去の対話文脈から作成された query/key 間の類似度を計算するために，それぞれの文脈を Sentence-BERT [15] によってベクトル化する．Sentence-BERT は，BERT [5] の出力部分に Pooling 層を加えて，文章埋め込みを出力するように変更したモデルである．

Sentence-BERT の学習では，文章（対話文脈）間の距離を学習するために，入力文章 (anchor)，類似する文章（正例），類似しない文章（負例）の triplet を学習事例として使用する．本研究では，同一セッション内の対話文脈は関連しやすいと仮定し，同一セッションに含まれる対話文脈間の距離が近く，異なるセッションに含まれる対話文脈間の距離が遠くなるように，anchor と正例は同一セッションから，また負例は anchor とは異なる（同じ話者間の）セッションから抽出して triplet を構成し，モデルの学習を行う．この際，triplet の各文章には同じターン数 (1-3) で同じ話者の発話で終了する文脈を使用する．これは，検索時に query と key の長さが乖離している場合，類似度を適切に算出することが難しいと考えたためである．

4.2 過去の対話文脈に即した生成器

過去の対話文脈を類似度昇順にソートし，その末尾で現在の対話文脈と結合した後，生成器の最大入力長 (256, 1024) を超える場合は，超過した部分を先頭から切り捨て，モデルに入力する．生成器は GPT-2 [14] の事前学習済み重みで初期化され，参照応答に対するクロスエントロピー損失が最小化されるように微調整する．現在の対話文脈を $\boldsymbol{x}$，N 個の過去の対話文脈を $\boldsymbol{v} = (v_1, ..., v_N)$，参照応答を $\boldsymbol{y}$ とすると，損失関数は次のように記述される．（ただし $\boldsymbol{y}$ のトークン数を $|\boldsymbol{y}|$ 個，$y_{1:t-1} = y_1, y_2, \ldots, y_{t-1}$ とする）

$$L_{model} = -\sum_{t=1}^{|\boldsymbol{y}|} \log p(y_t|\boldsymbol{v}, \boldsymbol{x}, y_{1:t-1})$$

話者を一貫して識別するための話者トークン [S1], [S2] を各発話の先頭に付与し，発話の終端を示すための </s> トークンを各発話の終端に加えた．また．過去の対話文脈間の境界を示すために，各対話文脈の終端に [EOH] トークンを付与し，現在と過去の対話文脈の境界には代わりに [CUR] トークンを付与した．

5 実験

本節では，提案手法の性能を評価するために，過去文脈抽出器を統合した GPT-2 ベースの対話モデルを，Twitter 長期間対話データセット (3 節) を用いて訓練し，自動評価と人手評価で性能を検証する．

5.1 モデル

各モデルは，PyTorch 1.10.2[3] および Transformers 4.20.0[4] を用いて実装した．抽出器に用いた Sentence-BERT [15] の学習データは Twitter 長期間対話データセット（3 節）を基に構築し，各エピソード（特定 2 話者についての対話セッション群）から 1 つの triplet を生成した．Sentence-BERT は，日本語版 BERT [5][5] に Pooling 層を加えて実装し，5 エポック訓練を行って，開発データでの分類精度が最良のモデルを採用した．

応答生成モデルには日本語版 GPT-2 [14][6] を使用し，過去と現在の対話文脈を入力して参照応答を生成するように学習を行った．ベースラインには，直前の過去セッションを時系列順に入力する手法 [22] を使用し，参考として過去の対話文脈を用いないモデルも学習を行った．提案手法については，4.1.1 節で記述した value（Session, Key, Next, KeyNext）の比較と，query/key の長さ（1-3 発話, 全 query 使用）の比較を行った．各モデルは patience を 1 とする early stopping を用いて最大 5 エポック微調整し，開発データでの損失が最小となるモデルについてそれぞれ性能を評価した．

また，訓練データにおいて応答生成モデルへの入力長が最大入力長を上回る事例が大多数を占める場合には，生成応答長の学習が安定しないことを確認した．そこで本実験では，応答の終端を学習するために，最大入力長を超過した分入力を切り捨てる処理に加えて，訓練データセット中で最長の参照応答が生成できるように，追加で入力の一部を切り捨てた．この時，入力文脈が極端に切り捨てられないように，参照応答長が上位 5% となるデータを除外し，モデルの訓練・評価を行った．

[3] https://pytorch.org/
[4] https://huggingface.co/
[5] https://huggingface.co/cl-tohoku/bert-base-japanese-whole-word-masking
[6] https://huggingface.co/rinna/japanese-gpt2-small

過去文脈	Values	PPL (↓)	BLEU-2/3 (↑)	DIST-1/2 (↑)	#values
なし		49.71	1.21/0.56	7.53/22.39	-
最大入力長：256					
直近の履歴	Session	45.53	1.99/0.82	5.03/19.09	1.70
提案手法	Session	45.62	1.89/0.76	5.07/19.63	1.61
	Key	47.34	1.75/0.76	4.97/18.19	7.69
	Next	**44.21**	**2.14/0.88**	5.31/20.46	8.03
	KeyNext	44.91	2.02/0.84	**5.60/21.41**	4.54
最大入力長：1024					
直近の履歴	Session	42.94	2.02/0.84	5.47/22.67	7.30
提案手法	Session	43.19	1.94/0.83	5.73/22.84	6.68
	Key	46.88	1.57/0.63	5.01/19.59	40.98
	Next	42.11	1.71/0.79	5.98/23.36	39.44
	KeyNext	**41.78**	**2.10/0.86**	**6.20/25.39**	25.25

表 2: 自動評価の結果（提案手法の抽出器で使用している query/key の長さは 1 発話としている）.

Query types	PPL (↓)	BLEU-2/3 (↑)	DIST-1/2 (↑)	#values
最大入力長: 256				
1 utt.	**44.91**	**2.02**/0.84	5.60/**21.41**	4.54
max 2 utts.	45.26	1.84/0.79	5.39/20.25	3.82
max 3 utts.	45.09	1.88/0.74	5.12/19.85	3.43
1+2+3 utts.	45.44	1.93/**0.85**	**5.71**/21.05	4.51
最大入力長: 1024				
1 utt.	**41.78**	2.10/0.86	**6.20/25.39**	25.25
max 2 utts.	42.52	2.09/0.86	5.52/22.92	20.07
max 3 utts.	42.11	2.12/**0.87**	5.75/23.48	17.28
1+2+3 utts.	41.94	**2.15**/0.82	5.81/24.22	25.05

表 3: query/key の種類ごとの比較 (提案手法では一貫して value に KeyNext を使用している).

5.2 結果: 自動評価

表 2 に，各手法の応答性能を自動評価した結果を示す．評価指標としては，Perplexity, BLEU-2/3 [13], DISTINCT-1/2 [10] を用い，生成応答の単語分割には，MeCab[7] (UniDic 2.1.2 辞書) を用いた．また，実際にモデルに入力された，過去の対話文脈の value の平均個数も測定した．提案手法の value に KeyNext を用いたモデルは，セッション単位で入力するベースラインに比べ，一貫して性能が高い．また，value に Next を使用したモデルは，最大入力長が 256 トークンの時はベースラインより性能が高いが，1024 トークンの時は性能が低下する．これは，value が Next の時は 1024 トークン以内にほぼ全ての value が収まり，有用性の低い過去の対話文脈まで参照されるためと考えられる．一方，Key や Session を value に用いる場合はベースラインに比べ性能が低いことから，Key やそれ以前の文章は，Key 以降の文に比べて有用ではないと予想される．また，過去の対話文脈を類似度順に入力するより，時系列順に入力する方が性能向上に効果的である可能性も考えられる．

[7] https://taku910.github.io/mecab/

モデル	Values	妥当性	文脈一貫性	人間らしさ
ベースライン	Session	-0.219	-0.0246	0.127
提案手法	KeyNext	**-0.107**	**0.0475**	**0.175**

表 4: 人手評価の結果（各被験者ごとに標準化されたスコアを基にした平均値）.

提案手法のうち KeyNext を value としたモデルを，query/key の種類別に比較した結果を表 3 に示す．max 2, 3 utt. では，現在の対話文脈から query を作成する際の最長発話数を 2, 3 とし，同じ長さの key を検索する．また 1+2+3 utts. では，1, 2, 3 発話 query/key を全て使用する．自動評価の結果，query/key の長さが 1 発話の場合の性能が一番高かった．これは，長い query/key を使用すると，query と高い類似度を持つ key が少なくなり，検索性能が低くなることで，有用な対話文脈を上手く抽出できなくなるためと考えられる．また，1+2+3 utts. の場合の性能は，1 発話 query/key を使用する場合とほぼ同じとなった．これは，1 発話 query/key 間の類似度が高くなりやすく，1 発話 key と紐付いた value が多く選択されるためだと考察される．

5.3 結果: 人手評価

人手評価として，3 人のアノテータがベースラインと提案手法の各生成応答を評価した．アノテータは 116 個の生成応答に対し，Ji ら [8] の手法に従い，以下 3 つの評価基準について 0-100 の整数値で点数をつけた．

妥当性: 直前の入力発話に対して妥当な応答か

文脈一貫性:（過去の対話セッションを含む）対話文脈と矛盾しない応答か

人間らしさ: 人間らしい応答か

過去セッション数が平均 15.49 であることを考慮し，15-17 個の過去セッションを含む 10 エピソードを評価データセットからサンプリングし，最終 3 セッションの生成応答を評価した．比較する手法をベースラインと最も性能が高い提案手法（query/key が 1 発話で，value を KeyNext としたモデル）とし，どちらも最大入力長を 1024 トークンとした．

表 4 に人手評価の結果を示す．ベースラインに比べ，提案手法では文脈一貫性において大きく改善しており，長期的な文脈を考慮し一貫した応答が生成できていることがわかる（t 検定時の p 値は $0.039 < 0.05$）．また，妥当性と人間らしさの観点では，提案手法は平均値においてベースラインを上回っているものの，t 検定時の p 値は 0.05 を上回っていた（0.105 と 0.111）．これは，妥当性の高い応答や人間らしい応答は，直前の発話のみあれば基本的に可能であり，参照する過去の発話の品質に大きく依存しないためだと考えられる．

6 まとめ

本論文では，長期的雑談対話において効果的に過去の対話文脈を使用する対話システムを提案した．提案手法では，現在の対話文脈を基に，応答生成に有用な過去の対話文脈を，セッションより小さい単位で検索・抽出する．提案手法で使用する抽出器は，対話データセットから構築されるデータで訓練可能であり，追加の教師情報を必要としない．Twitter 上の大規模対話ログから構築した，長期間に亘る対話データを用いた実験の結果，提案手法はセッション単位での抽出器に比べ，自動評価・人手評価で高い応答性能でよりあることがわかった．今後の課題としては，疎な注意機構を使用することで計算量を削減することや，value 内の各発話の役割を表す情報を与えることが挙げられる．

謝辞

この研究は国立情報学研究所（NII）CRIS と LINE 株式会社が推進する NII CRIS 共同研究，および JSPS 科研費 JP21H03494 の助成を受けています．人手評価にご協力くださった研究室の方々に深く感謝致します．

参考文献

[1] D. Adiwardana, M.-T. Luong, D. R. So, J. Hall, N. Fiedel, R. Thoppilan, Z. Yang, A. Kulshreshtha, G. Nemade, Y. Lu, and Q. V. Le. Towards a human-like open-domain chatbot, 2020.

[2] S. Bae, D. Kwak, S. Kang, M. Y. Lee, S. Kim, Y. Jeong, H. Kim, S.-W. Lee, W. Park, and N. Sung. Keep me updated! Memory management in long-term conversations. In *Findings of EMNLP*, 2022.

[3] T. W. Bickmore and R. W. Picard. Establishing and maintaining long-term human-computer relationships. *ACM Trans. Comput. Hum. Interact.*, 2005.

[4] D. Cai, Y. Wang, W. Bi, Z. Tu, X. Liu, and S. Shi. Retrieval-guided dialogue response generation via a matching-to-generation framework. In *EMNLP-IJCNLP*, 2019.

[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019.

[6] K. Guu, K. Lee, Z. Tung, P. Pasupat, and M.-W. Chang. REALM: Retrieval-augmented language model pre-training. In *ICML*, 2020.

[7] G. Izacard and E. Grave. Leveraging passage retrieval with generative models for open domain question answering. In *EACL*, 2021.

[8] T. Ji, Y. Graham, G. Jones, C. Lyu, and Q. Liu. Achieving reliable human assessment of open-domain dialogue systems. In *ACL*, 2022.

[9] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, and D. Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *NeurIPS*, 2020.

[10] J. Li, M. Galley, C. Brockett, J. Gao, and B. Dolan. A diversity-promoting objective function for neural conversation models. In *NAACL: HLT*, 2016.

[11] K. Nishida, N. Yoshinaga, and K. Nishida. Self-adaptive named entity recognition by retrieving unstructured knowledge, 2022.

[12] G. Pandey, D. Contractor, V. Kumar, and S. Joshi. Exemplar encoder-decoder for neural conversation generation. In *ACL*, 2018.

[13] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. Bleu: a method for automatic evaluation of machine translation. In *ACL*, 2002.

[14] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners, 2019.

[15] N. Reimers and I. Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *EMNLP-IJCNLP*, 2019.

[16] A. Ritter, C. Cherry, and W. B. Dolan. Data-driven response generation in social media. In *EMNLP*, 2011.

[17] S. Roller, E. Dinan, N. Goyal, D. Ju, M. Williamson, Y. Liu, J. Xu, M. Ott, E. M. Smith, Y.-L. Boureau, and J. Weston. Recipes for building an open-domain chatbot. In *EACL*, 2021.

[18] S. Sato, N. Yoshinaga, M. Toyoda, and M. Kitsuregawa. Modeling situations in neural chat bots. In *ACL SRW*, 2017.

[19] K. Shinzato, N. Yoshinaga, Y. Xia, and W.-T. Chen. Simple and effective knowledge-driven query expansion for QA-based product attribute extraction. In *ACL*, 2022.

[20] S. Wang, Y. Xu, Y. Fang, Y. Liu, S. Sun, R. Xu, C. Zhu, and M. Zeng. Training data is more valuable than you think: A simple and effective method by retrieving from training data. In *ACL*, 2022.

[21] Y. Wu, F. Wei, S. Huang, Y. Wang, Z. Li, and M. Zhou. Response generation by context-aware prototype editing. *AAAI*, 2019.

[22] J. Xu, A. Szlam, and J. Weston. Beyond goldfish memory: Long-term open-domain conversation. In *ACL*, 2022.

[23] X. Xu, Z. Gou, W. Wu, Z.-Y. Niu, H. Wu, H. Wang, and S. Wang. Long time no see! open-domain conversation with long-term persona memory. In *Findings of ACL*, 2022.

[24] S. Zhang, E. Dinan, J. Urbanek, A. Szlam, D. Kiela, and J. Weston. Personalizing dialogue agents: I have a dog, do you have pets too? In *ACL*, 2018.

[25] Y. Zhang, S. Sun, M. Galley, Y.-C. Chen, C. Brockett, X. Gao, J. Gao, J. Liu, and B. Dolan. DialoGPT: Large-scale generative pre-training for conversational response generation. In *ACL, System Demonstrations*, 2020.