質問・応答集からの文選択による雑談対話システムの学習

太刀岡 勇気
デンソーアイティーラボラトリ
ytachioka@d-itlab.co.jp

欅 惇志
デンソーアイティーラボラトリ
akeyaki@d-itlab.co.jp

1 はじめに

ユーザからの質問に応答するチャットボットのような対話システムでは，大量の質問・応答ペアを用意してユーザの質問と用意した質問との類似度に基づいて対応する応答を選択する用例ベースの手法がよく利用される [1]．近年ではこのような方法がある程度実用化されてきたことから，質問・応答だけでない，雑談のような，より難しい非タスク指向型対話システムの研究も盛んである．その場合，ルールを記述することはほぼ不可能であり，ユーザの発話は予見不能であることから，すべての対応関係を収集することは難しい．そこで seq2seq の手法 [2] が良く用いられる．seq2seq は，ユーザの発話に対するシステムの応答を翻訳問題として結びつけるもの [3] であり，どのようなユーザ発話に対しても応答を生成できるという利点がある一方で，学習に大量のペアデータが必要であるという問題もある．大量のテキストデータが利用可能な資源としては，twitter や『おーぷん 2 ちゃんねる対話コーパス [4]』がある．ただし，これをそのまま対話エージェントの応答生成に適用しようとすると，文のスタイルが適さない点が問題になる[1)]．対話エージェントには，特に商用に用いる場合，ある程度フォーマルで丁寧な応答が求められるからである．

これに対して，「Yahoo!知恵袋」等の質問・応答集を使うと，丁寧に書かれていることが多いため，スタイルの問題は起こりにくい．さらに，すでに集めてある質問・応答集の再利用ができる．しかし，質問・応答ともに長文であることが多いため，seq2seq の学習が難しいことに加え，生成される応答が長すぎるという問題がある．例えば後に示すように『Yahoo!知恵袋』では，100 文字程度の回答が多いが，音声での対応を考えた場合，エージェントが 1 分間に話せるのは 300 文字程度なので，100 文字発話するのには 20 秒程度かかる．質問に対する必要な情報提供であれば適切な長さであるが，雑談としては応答が 20 秒続くとかなり長く感じられる．また，現状の seq2seq 方式での生成では，それだけの長文を理路整然とさせるのは難しい．意味不明な文や自己矛盾した文が生成される可能性も高く，これはユーザの満足度を著しく低下させる．そこで，質問・応答集に対して適切な文選択を行い，ペアデータを短い文で構築することで，この問題を解決することを試みる．選択法はいくつか考えられるため，ここでは可能な選択法を 7 種に関して，seq2seq による応答文生成のパープレキシティと生成された応答文長の変化を検討した．

2 seq2seq に基づく応答生成

2.1 手法概要

翻訳での成功 [2] を受けて，対話でもソース (src) からターゲット (tgt) への変換をニューラルネットで直接学習する seq2seq に基づく応答生成 [3] が用いられてきており，雑談応答生成への適用も行われている [5, 6]．通常入力は文字単位ではなく，単語やそれに類するまとまりの単位とする．いくつかの方式が提案されているが，ここでは opennmt[7] で使われている固定次元の埋め込みベクトルに基づき LSTM で encode-decode する方式を採用した．

2.2 係り受け解析による冗長箇所の除去

seq2seq による応答文には，応答が相槌のような無難なものになりやすいという問題が指摘されている [5, 6]．また同一の語句の繰り返しが多くみられるという問題がある[2)]．例えば「ステーキのステーキは，肉が柔らかくて美味しいですよね」というような過

1） 特に後者では独特の関西弁風のスタイルが問題となる．

2） 特に学習に用いるペア文の文長が長くなると，生成される応答文に繰り返しが増える傾向がみられた．

度に冗長な文が頻繁に生成され，これがユーザの満足度を著しく低下させる．そこで，係り受け解析を行い，係元が係先と同じ名詞を含む場合は係元を削除することで冗長箇所を除去することとする．先の文に対しては，「ステーキの」が「ステーキは」にかかっているので，「ステーキの」を削除する．

3 質問・応答集からの文選択

提案法では，質問・応答集を「. ?」で文に分割し，そこから対象の文を選択することで，情報をなるべく失わずに，ペアの文の長さを短くする．文選択には，以下の 7 種を検討した．

1. allpairs: 質問文と応答文の全組み合わせ
2. crosspairs: 質問文と応答文のはじめの文と終わりの文の最大 4 通りの組み合わせ
3. crosspairs2: 質問文と応答文のはじめの文とはじめの文，質問文の終わりの文と応答文のはじめの文の最大 2 通りの組み合わせ (下に示す first-first と last-first の組み合わせ)
4. longest: 最大の文字数の文同士の組み合わせ
5. last-first: 質問文の終わりの文と応答文のはじめの文の組み合わせ
6. first-first: 質問文と応答文のはじめの文とはじめの文の組み合わせ
7. first-first & longest: first-first と longest の組み合わせ

allpairs は，すべての組み合わせを考えるもので，例えば質問・応答のペアが src が 2 文，tgt が 3 文からなる場合，この 1 ペアから 6(= 2 × 3) 通りのペアデータを作成する．質問文と応答文に関しては中間の文はあまり重要ではなく，はじめの文と終わりの文が重要であると考えられる[3]ので，それらの組み合わせをとる crosspair と，応答文についてははじめの文のみを追加した crosspair2 を検討した[4]．また短い文は挨拶や「ありがとうございます」のように情報が少ないことが多いため，最大の長さの文を組み合わせる方法も考えられる (longest)．さらには crosspair2 を別々にした last-first と first-first を検討した．また first-first と longest を組み合わせた方法についても検討した．3 手法 (first-first,last-first,longest) 以外はペア数を増やすことができるため，学習データを拡大する効果も得られる可能性がある．

3） 特に質問文に関しては，はじめの文で「～は～ですか」のように質問したあとで「ちなみに～です」のように補足を行う場合 (はじめの文が重要) と，「16 歳です．～は～ですか」のように自分の前提を述べた後に質問する場合 (終わりの文が重要) の 2 種類が典型的である．

4） 応答文に関しても「通りすがりのものです．～です」のように，はじめの文に情報がない場合もあるが，質問文よりははじめの文が重要である比率が高いため，応答文に関しては，はじめの文だけとした場合も検討した．

4 実験

4.1 実験条件

seq2seq は大量のペアデータから学習する必要があるため，ベースのモデルは『おーぷん 2 ちゃんねる対話コーパス [4](以下 open2ch)』を用いて構築した[5]．ベースモデルの学習後，質問・応答集でファインチューニングする．質問・応答集には，『現代日本語書き言葉均衡コーパス (以下 bccwj) [8]』に含まれる「Yahoo!知恵袋 (OC)」を利用した．OC から差別や性的な内容等を含む不適切な文を除外するため，[4] で行われているフィルタリング処理を行い，最終的に 5.4 万ペアが得られた．さらに，slack 上で収集した『テキスト雑談コーパス (以下 slack)[9, 10]』を用いた．発話ペア数は 1.4 万と少ないが，丁寧な口語調のデータである．表 1 に学習に用いたペア数と平均文字数を示す．open2ch に比べると，bccwj と slack のペア数がかなり少ないことと，文長が 3～4 倍程度とかなり長いことがわかる．

opennmt [7] を用いて seq2seq モデルを構築した．前処理に sentencepiece[11] を用いて 16,000 ピースにより，文を分割した．これを 500 次元の埋め込みベクトルに変換し，LSTM2 層により encode し，LSTM と attention を用いて decode を行う．ビームサーチにより最終的な応答文を得る．open2ch の学習は 20 万回[6]繰り返し行った．これに対して，7 万回 open2ch で学習したモデルを初期モデルとして，学習率を低減させて bccwj+slack により最大 5 万回ファインチューニングを行う．提案の文選択を行った場合を同じく表 1 の 3 段目に示す．longest を除き，open2ch とほぼ同等な平均文長を得ることができた[7]．また異なる組み合わせを利用することでペア数を増やすことができている．ただし (特に allpairs では)，異なる質問に対して同一の応答が対応する，あるいは，同一の質問に対して異なる応答が対応する例が多い．

5） 「サンガツ」「ワイ」等の open2ch に特有ないくつかの表現は，「ありがとう」「私」などの対応する語句と置き換えた．

6） エポックではなく，バッチ単位での繰り返し数である．

7） 応答文は若干長い．

表 1　学習とテストに用いたペア数と平均文字数．検証データは別に用意

	ペア数	平均文字数	
		src	tgt
open2ch	6,937,598	26.7	22.7
open2ch (test)	12,270	26.7	22.8
bccwj (OC)	54,089	78.2	98.2
slack	14,449	68.7	67.3
bccwj+slack (=baseline)	68,538	76.2	91.7
bccwj+slack (test)	1,999	77.4	102.4
allpairs	562,642	30.1	31.1
crosspairs	196,333	30.7	31.4
crosspairs2	112,932	30.9	32.1
longest	68,539	43.0	45.4
last-first	68,539	30.6	32.0
first-first	68,539	33.9	32.1
first-first & longest	113,271	38.1	38.1

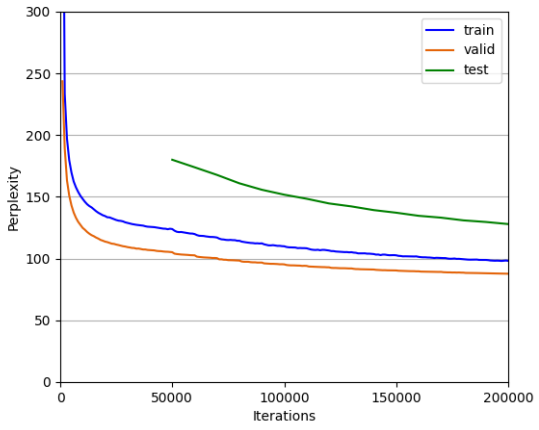

図 1　open2ch データベース学習時のパープレキシティ

4.2　結果 (パープレキシティでの比較)

図 1 に，open2ch で学習した際のパープレキシティを示す[8]．学習 (train)，検証 (valid)，テスト (test) のすべてで繰り返しに伴い単調に減少している．

図 2 には，質問・応答集 (bccwj+slack) でファインチューニングした際のパープレキシティを示す．検証は各手法に合わせて文選択を行ったもので評価した．テストは文選択をしない bccwj+slack の 1,999 ペアで行っているため，baseline のテストのみ学習とマッチした条件で，それ以外はミスマッチな条件になっている．そのため，すべての場合でテストのパープレキシティは baseline の場合に比べて悪化している．

個別にみると，まず，すべての組み合わせを使う

8）　学習のパープレキシティは直近 1,000 回の平均を 1,000 回ごと，検証とテストのパープレキシティはそれぞれ 1,000 回，5,000 回ごとに測定した．以降も同一である．

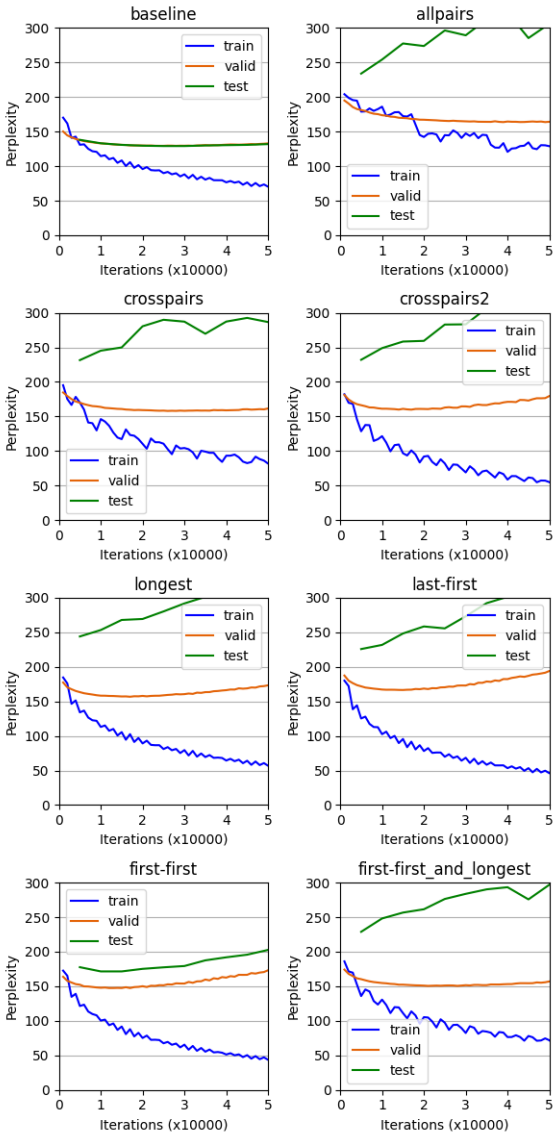

図 2　質問・応答集 (bccwj+slack) でファインチューニング時のパープレキシティ

allpairs は学習のパープレキシティも十分に低下せず最も悪い結果となった．これは上述の通り，同一の (異なる) 質問に対して異なる (同一の) 応答が対応することで学習がうまくいかなかったためと考えられる．crosspairs に関しても同様の理由で学習のパープレキシティが単調減少せず，期待された学習データ拡大の効果は得られなかった．最も検証のパープレキシティが小さくなったのは，first-first の 1 万〜2 万回のファインチューニングを行った場合であった．テストのパープレキシティでも first-first が最もよく，ミスマッチ条件にもかかわらず，マッチ条件の baseline と近い値となっている．first-first に longest を足すと検証のパープレキシティは安定したが，テストセットのパープレキシティは低下し

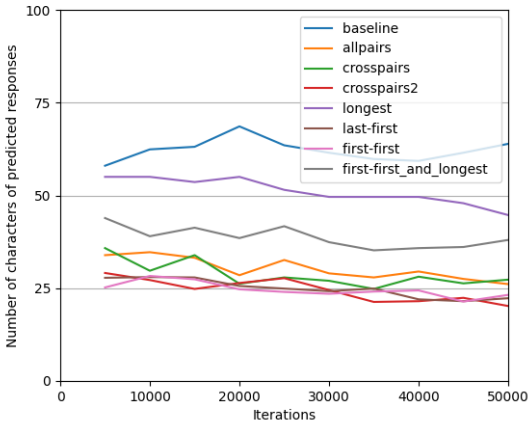

図 3 テストセット (bccwj+slack) の src に対して生成された応答文の平均文字数

た．これから文選択は first-first が最もよいと考えられる．

4.3 結果 (文長での比較)

図 3 に，テストセット (bccwj+slack) に対して生成された応答文の平均文字数を示す．テストセットは，表 1 に示した通り，src が 77.4 文字，target が 102.4 文字であった．全体として，ファインチューニングの繰り返し回数によらず，ほぼ文選択法により文長が決まっていることがわかる．個別にみると，baseline はかなり長く平均で 60 文字程度の応答を生成していた．それから longest，first-first_and_longest と続く．これらが平均で約 40 文字を超えている．一方，それ以外の方法では 40 文字以下となっていて，open2ch での文長と同等であり，first-first か crosspair2(first-first を含む) が最も文長が短くなっている．first-first による文選択では，元の文の情報低下を抑えながら (パープレキシティは維持しながら)，文長を短くできているといえる．

4.4 結果 (応答例と冗長箇所除去の効果)

雑談を想定したユーザ発話 (上記テストセットには含まれない) に対する応答例を図 4 に示す．はじめに述べた通り，open2ch の応答はエージェントとしてはふさわしくなく，これに限らず回答の多くが，「まじかよ…」，「ありがとう!(もともとは「サンガツ!」)」，「そうなんか」になってしまう．baseline はスタイルはよいが冗長な出力が出ている．ただし「ステーキのステーキは，...」に関しては，冗長箇所除去を行うことでかっこ内の文に変換できた[9]．提

```
%open2ch
先週は神戸に行ったよ > まじかよ…
明日は何しよっか > おーぷん民
昨日はステーキを食べたよ > ありがとう!
ケーキを作ってみようと思って作り方を調べた > そうなんか
%baseline
先週は神戸に行ったよ > 神戸に行ったことないなぁ
明日は何しよっか > 明日に備える
昨日はステーキを食べたよ > ステーキのステーキは，肉を食うのが一般的です (=ステーキは，肉を食うのが一般的です)
ケーキを作ってみようと思って作り方を調べた > ケーキやケーキ，ケーキ，ケーキ，お菓子，(...ケーキ，お菓子を繰り返す)
%first-first
先週は神戸に行ったよ > 私も神戸に行ったことがあります
明日は何しよっか > 明日休みます
昨日はステーキを食べたよ > 私はステーキを食べます
ケーキを作ってみようと思って作り方を調べた > ケーキを作るのは簡単ですね
```

図 4 ユーザ発話 (‘>’ の左) に対する生成された応答 (‘>’ の右) の例

案の first-first は，短い文で比較的適切な応答を返せている．

5 まとめと今後の課題

雑談対話システムの学習に，大量のデータがある質問・応答集を利用する場合，文長が長いためモデルの学習が難しく，生成される文が長くなりすぎるという問題があるため，文選択を行うことで文長を制限する方法を提案した．7 種の異なる文選択法を検討した結果，はじめの文同士を組み合わせたものが最もパープレキシティが低く，文長も短くできることがわかった．今回は比較的単純な方法で文選択を行ったが，今後，重要文の推定に基づき，主題となる文の選択を行った場合との比較を行う予定である．

9) 係り受け解析には，CaboCha[12] を利用した．

参考文献

[1] 東中竜一郎, 稲葉通将, 水上雅博. Python でつくる対話システム. オーム社, 2020.

[2] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Proc. NIPS*, 2014.

[3] Oriol Vinyals and Quoc V Le. A neural conversational model. In *Proc. ICML*, 2015.

[4] 稲葉通将. おーぷん 2 ちゃんねる対話コーパスを用いた用例ベース対話システム. 第 87 回言語・音声理解と対話処理研究会 (第 10 回対話システムシンポジウム), 人工知能学会研究会資料 SIG-SLUD-B902-33, pp. 129–132, 2019.

[5] Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. A diversity-promoting objective function for neural conversation models. In *Proc. NAACL-HLT*, pp. 110–119, 2016.

[6] 高山隼矢, 荒瀬由紀. Distant supervision を用いた情報量を制御可能な雑談応答生成. 言語処理学会 第 26 回年次大会 発表論文集, pp. 323–326, 3 2020.

[7] Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. Opennmt: Open-source toolkit for neural machine translation. In *Proc. ACL*, 2017.

[8] Kikuo Maekawa, Makoto Yamazaki, Toshinobu Ogiso, Takehiko Maruyama, Hideki Ogura, Wakako Kashino, Hanae Koiso, Masaya Yamaguchi, Makiro Tanaka, and Yasuharu Den. Balanced corpus of contemporary written japanese. *Language Resources and Evaluation*, Vol. 48, No. 2, pp. 345–371, 6 2014.

[9] Hiroshi Tsukahara and Kei Uchiumi. System utterane generation by label propagation over association graph of words and utterance patterns for opendomain dialogue systems. In *Proc. PACLIC-29*, 2015.

[10] 塚原裕史, 内海慶. 言い換えを利用した対話行為推定の汎化性能向上. 言語処理学会 第 22 回年次大会 発表論文集, pp. 87–90, 3 2016.

[11] Taku Kudo and John Richardson. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proc. Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 66–71, Brussels, Belgium, 11 2018. Association for Computational Linguistics.

[12] 工藤拓, 松本裕治. チャンキングの段階適用による日本語係り受け解析. 情報処理学会論文誌, Vol. 43, No. 6, pp. 1834–1842, 2002.