質問-回答対を利用した半教師有り抽出型質問要約

町田 和哉 [1]　小林 隼人 [2,3]　高村 大也 [4,5]　奥村 学 [5]
[1] 東京工業大学大学院　[2] ヤフー株式会社　[3] 理化学研究所 AIP センター
[4] 産業技術総合研究所　[5] 東京工業大学科学技術創成研究院
{machida, takamura, oku}@lr.pi.titech.ac.jp hakobaya@yahoo-corp.jp

1 はじめに

Yahoo!知恵袋[1]などに代表されるコミュニティQA サイトでは，検索によって目的の質問を探すことになるが，検索結果には表示領域の制約から質問文書全文ではなく質問文書の先頭が見出し文として表示される (図 1)．しかし，この見出し文では投稿者の意図した質問内容の理解に不十分な場合があり，先頭ではなく質問文書中で最も重要な文が表示されることが望ましい．そこで，本研究では重要文を表示するために抽出型の質問文書要約に取り組む．

近年の抽出型要約 [1, 2, 3] は，ニューラルネットワークによる手法が主流となり，大量の正解抽出文ラベル付きデータを教師データに用いて学習を行っている．しかし，常に大量のラベル付き教師データが用意できるとは限らず，作成には高いコストがかかってしまう．この課題を解決するため，我々はコストをかけず大量に入手できる質問-回答対データに注目し，補助的に利用することで要約性能の向上を目指す．

本研究では，質問-回答対データを利用する際に，Encoder-decoder モデル [4] をベースとしたモデルを用い，各入力文に対して重要文判定と回答文生成を同時に学習するモデルを提案する．モデルが出力する抽出文は，重要文判定でのスコアと回答文生成時に計算される各文への重みを組み合わせて選択される．この重みは回答文を出力する際にどの入力文を注視しているか，つまり，質問に答えるためにはどの文に注目するべきかという重要度と捉えることができ，重要文判定の手がかりとなることが期待できる．人手で作成した評価セットによる比較実験において，提案手法がベースラインよりも要約に含めるべき文を正しく判定できたことを報告する．

本研究の貢献は以下の通りである．

- 質問文書における抽出型要約において，回答文の利用が性能向上に寄与することを示した．
- 異なる学習方法のモデルで実験を行い，回答文の効果的な利用方法についての知見を得た．
- 質問文書における正解抽出文データセットを作成した．

[1]https://chiebukuro.yahoo.co.jp/

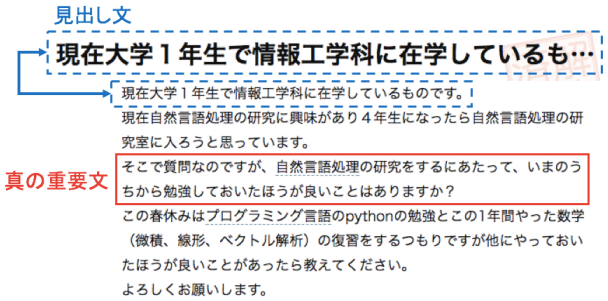

図 1: Yahoo!知恵袋に投稿された質問例

2 関連研究

要約に関する研究は幅広く取り組まれているが，その多くは新聞記事や学術論文などに関するものであり，質問文書に対して要約技術を適用したものは少ない．

Ishigaki ら [5] は，タイトル付きで質問が投稿されるコミュニティQA サイトに注目し，質問とそのタイトルの対を長文質問とその要約の対とみなし，抽出型および生成型の要約モデルを学習した．Tamura ら [6] は，質問応答システムの性能を向上させる目的で，複数文で構成される質問から核文を同定する分類器を学習した．この研究では質問タイプと核文を人手でアノテートした質問文書データが用いられている．Higurashi ら [7] は，クラウドソーシングを利用して質問文書の見出しを付与した教師データを作成し，それを用いたランク学習による見出し生成に取り組んでいる．石垣ら [8] は，単一文質問（一文で構成される質問）と要約に含めるべき文が似た特徴を持つことに着目し，単一文質問を正例，10 文以上で構成される質問中の文を負例として教師データを構築し，Distant supervision による抽出型質問要約器を提案している．これらの研究は質問文書のみを扱っており，質問-回答対データを利用する点で本研究とは異なる．

3 提案手法

Encoder-decoder モデルをベースとして質問-回答対データを利用する抽出型要約モデルの概要を図 2 に示す．本研究ではエンコーダとデコーダに LSTM[9] を用いる．このモデルは，入力文書に対して，重要文判定と回答文生成を同時に学習する．入力文書 d は文の系列 $d = (s_1, s_2, ..., s_m)$，文 s_i は単語系列 $s_i = (w_{i,1}, w_{i,2}, ..., w_{i,n})$ である．このモデルは，階層的なエンコーダを適用しており，単語エンコーダで各文の単語系列をエンコードして文の隠れ状態 $\{h_1, h_2, ..., h_m\}$ を取得し，文エンコーダで文の隠れ状態をエンコードし文書ベクトル c を得る．エンコード後は，重要文判定と回答文生成 2 つのコンポーネントで処理を行い，抽出する文を選択する．

3.1 重要文判定

重要文判定では，各入力文が重要文かどうかの分類を行う．各入力文の隠れ状態に対して重要文である確率値を算出する：

$$p(s_i) = (p_{i,0}, p_{i,1}) = \mathrm{softmax}(W_{\mathrm{imp}} h_i). \quad (1)$$

$p_{i,0}, p_{i,1}$ はそれぞれ重要文でない確率，重要文である確率を表し，$W_{\mathrm{imp}} \in R^{2\times H}$ は重み行列であり，H は隠れ状態の次元である．また，ここでの損失関数は，次式の交差エントロピーで計算する：

$$loss_{\mathrm{imp}} = -\sum_{i=1}^{m} \{(1 - t_i) \log p_{i,0} + t_i \log p_{i,1}\}. \quad (2)$$

ここで，t_i は文 i の正解抽出文ラベルを表し，重要文であれば 1，重要文でなければ 0 の値を取る．

本研究では $p_{i,1}$ を重要文らしさの確信度とみなし，重要文スコア $score_{\mathrm{imp}}$ と定義する：

$$score_{\mathrm{imp}}(s_i) = p_{i,1}. \quad (3)$$

3.2 回答文生成

回答文生成では，単語デコーダで回答文を成す単語系列 $(y_1, y_2, ..., y_k)$ を出力する．単語デコーダの隠れ状態は初期状態を $h'_0 = c$ とし，デコード時刻 j における単語デコーダの隠れ状態 h'_j は直前の出力 y_{j-1} と隠れ状態 h'_{j-1} から次式によって計算される：

$$h'_j = \mathrm{LSTM}(h'_{j-1}, y_{j-1}). \quad (4)$$

単語の出力確率は，単語デコーダの隠れ状態 h'_j を用いて次式で計算される：

$$p(y_j | y_{<j}, d) = \mathrm{softmax}(W_o h'_j). \quad (5)$$

だたし，$W_o \in R^{V\times H}$ は重み行列であり，V は語彙サイズを表す．

ここでの損失関数には，負の対数尤度を用いる：

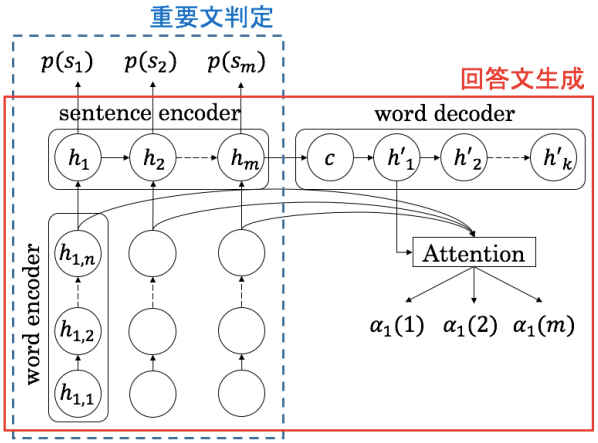

図 2: モデルの概要図

$$loss_{\mathrm{ans}} = -\sum_{(d,y)\in D} \log p(y|d). \quad (6)$$

D は訓練データセットを表す．

最終的なモデル全体の損失関数は，λ をハイパーパラメータとして $loss_{\mathrm{imp}}$ と $loss_{\mathrm{ans}}$ を組み合わせたものを最小化する：

$$loss = \lambda * loss_{\mathrm{imp}} + (1 - \lambda) * loss_{\mathrm{ans}}. \quad (7)$$

また，図 2 にあるように回答文生成時には，注意機構 (Attention)[10] を用いて入力文の隠れ状態から文脈ベクトルを獲得し単語予測に用いる．デコード時刻 j における文脈ベクトル c_j は入力文の隠れ状態の加重平均として次式で計算される：

$$c_j = \sum_{i=1}^{m} \alpha_j(i) h_i. \quad (8)$$

ここで重み $\alpha_j(i)$ は単語デコーダの時点 j における入力文の隠れ状態 h_i の重要度を表し，次式で計算される：

$$\alpha_j(i) = \frac{\exp(h'_j \cdot h_i)}{\sum_{i'=1}^{m} \exp(h'_j \cdot h_{i'})}. \quad (9)$$

注意機構を用いた場合には，次式で計算される $\hat{h}$ を式 (5) における h'_j として計算する：

$$\hat{h} = W_c[h'_j; c_j]. \quad (10)$$

$W_c \in R^{H\times 2H}$ は重み行列である．

各文に対して得られる式 (9) の重みを全ての出力単語について加算平均したものを回答文生成スコア $score_{\mathrm{ans}}$ と定義する：

$$score_{\mathrm{ans}}(s_i) = \frac{\sum_{j=1}^{k} \alpha_j(i)}{k}. \quad (11)$$

3.3 抽出文の選択方法

モデルの出力する抽出文は，重要文スコアと回答文生成スコアを組み合わせてスコアが最大の文を出力する．各文のスコアは κ をハイパーパラメータとして以下のように算出する：

$$score(s_i) = \kappa * score_{\mathrm{imp}}(s_i) + (1-\kappa) * score_{\mathrm{ans}}(s_i). \quad (12)$$

しかし，質問文書中には複数の質問事項が含まれる場合があり，どの質問事項がより重要であるか判定するのは難しい．本研究では，複数の質問事項が含まれる場合には，より先頭に出現する質問事項を出力するようにした．具体的には，算出されたスコアに対し，閾値 μ 以上のスコアを持つ文が複数存在する場合には，より先頭に出現する文を出力する．

4 実験

4.1 データセット

本研究では，実験データとして，Yahoo!知恵袋データセット第 1 版[2]を用いた．このデータには 2004 年 4 月から 2005 年 10 月に収集した 300 万件の質問と 1,300 万件の回答が含まれる．1 つの質問に対して複数の回答が存在するが，ベストアンサーのみ抽出することで，質問-回答対データを用意した．このデータに対して，句読点，‘?’，‘!’を文の境界として文分割を施し，各文は MeCab[3]で分かち書きした．加えて，質問文書について文数分布の統計情報を調べ，複数文で構成される質問のうち，95%以上を網羅する 7 文を最大文数として，8 文以上で構成される質問をデータから除いた．また, 文長が極端に長い事例を除くため 50 単語以上から成る文を含む事例を除いている．

4.2 正解抽出文データセット

質問-回答対データには正解抽出文ラベルが付与されていないため，正解抽出文ラベルの付いたデータセットを作成した．このデータセットの作成においては，質問-回答対データから 2 文以上で構成される事例を 2,000 件選び，質問文書に対して人手で正解抽出文ラベルの付与を行った．この作業にはクラウドソーシングサービス Lancers[4]を用いて 1 事例につき 5 人の作業者に要約に含めるべき文を選択してもらった．具体的な作業の手順は以下の通りである．

1. 質問文書全体に目を通し，質問内容を把握する．
2. 質問内容を理解するために最も必要な 1 文を選ぶ．
3. 1 文だけでは理解できない場合は追加で補足文を選ぶ．

4 人以上の作業者が要約に含めるべきと判定した文を正解抽出文とみなした．2,000 件の事例のうち，1 文のみが抽出文となった 1,180 文書を正解抽出文データとした．1,180 文書中の文数の分布を表 1 に示す．

[2]http://www.nii.ac.jp/dsc/idr/en/yahoo/yahoo.html
[3]http://taku910.github.io/mecab/
[4]http://lancers.co.jp

表 1: 重要文抽出データの事例数と文数の分布

文数	2	3	4	5	6	7
事例数	461	349	199	91	42	38

4.3 評価方法

性能評価は，式 (13) に示す正解率を文数ごとに計算し平均するマクロ平均で行う：

$$\text{正解率} = \frac{\text{抽出文を正しく同定できた文書数}}{\text{評価に用いた文書数}}. \quad (13)$$

正解抽出文データを 5 分割し，学習セット：3，開発セット：1，テストセット：1 として，5 分割交差検定のスコアの平均をモデルの性能評価に用いる．

4.4 実験設定

重要文判定のラベルのみで学習する手法をベースラインとして提案手法との比較を行った．提案手法については学習の方法が異なる 4 つのモデルを用意した．

$\mathbf{score}_{\mathrm{imp}}$ only: 重要文判定のみを学習する ($\lambda = 1.0$) ベースライン．このモデルは回答文を利用しない．抽出文の選択には，重要文スコアを使用する ($\kappa = 1.0$)．

$\mathbf{score}_{\mathrm{ans}}$ only: 重要文判定での学習は行わず，回答文生成のみを学習する ($\lambda = 0$)．抽出文の選択には，回答文生成スコアを使用する ($\kappa = 0$)．

separate learning: 重要文判定モデル $\mathrm{score}_{\mathrm{imp}}$ only ($\lambda = 1.0$) と回答文生成モデル $\mathrm{score}_{\mathrm{ans}}$ only($\lambda = 0$) を独立に学習させ，2 つのモデルが出力する重要文スコアと回答文生成スコアを抽出文の選択に使用する．κ は開発セットで決定する．

pre-train encoder: 質問-回答対データで回答文生成を事前学習し ($\lambda = 0$)，エンコーダの隠れ状態を更新後，重要文判定を学習する ($\lambda = 1.0$)．抽出文の選択には，重要文スコアを使用する ($\kappa = 1.0$)．

multi-task: 重要文判定と回答文生成を同時に学習する．λ，κ は開発セットで決定する．

これらのモデルに対して，表 2 に示す 3 種類の実験設定で性能評価を行った．multi-task において $\mathrm{L}_{\mathrm{small}}$+$\mathrm{U}_{\mathrm{large}}$ を訓練データとして学習する際には，不均衡データに対する工夫として，$\mathrm{L}_{\mathrm{small}}$ を重複して利用するオーバーサンプリングによる方法と $\mathrm{U}_{\mathrm{large}}$ を減らしてデータの比率を合わせるアンダーサンプリングによる方法で実験を行い，より学習の進むアンダーサンプリングの方法を採用した[5]．

モデルの実装には Chainer[6]を使用し，単語埋め込み層，エンコーダの隠れ層，デコーダの隠れ層の次元 (H) を 256，語彙サイズ (V) を 30,000 とした．単語

[5]ラベルなし質問-回答対データを 708 件ランダムにサンプリングしデータの比率を 1：1 に合わせた．
[6]https://chainer.org/

表 2: 実験設定

訓練データ	内容	事例数
正解ラベル付き質問-回答対データ (少量) L_{small}	正解抽出文ラベルの付いた質問-回答データ. 5 分割したうちの学習セットを用いる.	708
正解ラベル付き質問-回答対データ (少量) +ラベルなし質問-回答対データ (大量) L_{small}+U_{large}	上の正解ラベル付き質問-回答対データにラベルの付いていない大量の質問-回答対データを加えたデータ.	708+30 万
擬似ラベル付き質問-回答対データ (大量) L'_{large}	石垣ら [8] の提案した distant supervision による要約器が出力する抽出文を正解抽出文とみなしラベル付けしたデータ. ラベルなし質問-回答対データに擬似正解抽出文ラベルを付与している.	30 万

埋め込み層には知恵袋データで学習した word2vec の分散表現を初期値として設定し，ハイパーパラメータ λ，κ，μ は開発セットで 0 から 1.0 まで 0.1 刻みで探索を行い，最も正解率が高い値をテストセットでの設定とした．

4.5 実験結果

提案手法とベースラインの結果を表 3 に示す．どの設定においても multi-task が最も正解率が高く，ベースライン $score_{imp}$ only を上回った．これは回答文の情報が質問事項を捉えるための手がかりになっているためだと考えられる．各モデルの結果を見ると，$score_{ans}$ only は低い正解率となっており，回答文生成スコアのみでは効果が見込めないことが分かる．2 つのモデルを独立に学習する separate leraning はどのデータでも 70%を下回る正解率となり，multi-task のようにベースラインを超える結果にはならなかった．これは独立に学習したことで，重要文判定と回答文生成の学習が干渉し合うことがなく別々に最適化されてしまったことが原因と考えられる．pre-train encoder ではデータごとに傾向が異なる結果となった．表 3 に示すように，L_{small} ではベースラインと同等の正解率，L_{small}+U_{large} では上回る正解率，L'_{large} では下回る正解率となっている．事前学習することで質問文を捉えやすくなり重要文判定の精度が上がることが期待されるが，L_{small} ではデータサイズが小さいため事前学習の効果が薄く，L'_{large} では事前学習を行なったことで重要文判定が学習しやすくなる反面，擬似正解抽出文ラベルを過学習してしまったことが原因として考えられる．最も正解率が高い multi-task においては，各設定で $score_{imp}$ only との差を符号検定し，統計的有意 ($p < 0.05$) であることを確認した．このモデルでは重要文判定と回答文生成を同時に学習することにより，重要文スコアでは捉えられない事例に対して回答文スコアを組み合わせることで捉えることができるようになった．その中でも L'_{large} において高い正解率を示しており，擬似的な抽出文ラベルであっても大量にデータが用意できる場合，学習に効果的であることが分かる．

表 3: 各モデルでの実験結果

提案手法	L_{small}	L_{small}+U_{large}	L'_{large}
$score_{imp}$ only	.733	.733(同左)	.772
$score_{ans}$ only	.439	.663	.663(同左)
separate learning	.664	.680	.670
pre-train encoder	.734	.749	.750
multi-task	**.751**	**.761**	**.818**

5 おわりに

本研究では，質問文書に対する抽出型要約において，質問-回答対データを利用する手法を提案した．実験では，質問-回答対データの利用の有無で比較を行い，回答文を利用した場合の結果が利用しない場合の結果より高い正解率であることを確認した．今後の課題としては，1 文抽出ではなく複数文抽出するモデルへの拡張，生成型要約でも質問-回答対データの利用に効果があるかどうかの検証などが考えられる．

参考文献

[1] Mikael Kågebäck, Olof Mogren, Nina Tahmasebi, and Devdatt Dubhashi. Extractive summarization using continuous vector space models. In *Proceedings of CVSC*, pp. 31–39, 2014.

[2] Jianpeng Cheng and Mirella Lapata. Neural summarization by extracting sentences and words. In *Proceedings of ACL*, pp. 484–494, 2016.

[3] Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *Proceedings of AAAI*, pp. 3075–3081, 2017.

[4] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Proceedings of NIPS*, pp. 3104–3112, 2014.

[5] Tatsuya Ishigaki, Hiroya Takamura, and Manabu Okumura. Summarizing lengthy questions. In *Proceedings of IJCNLP*, pp. 792–800, 2017.

[6] Akihiro Tamura, Hiroya Takamura, and Manabu Okumura. Classification of multiple-sentence questions. In *Proceedings of IJCNLP*, pp. 426–437, 2005.

[7] Tatsuru Higurashi, Hayato Kobayashi, Takeshi Masuyama, and Kazuma Murao. Extractive headline generation based on learning to rank for community question answering. In *Proceedings of COLING*, pp. 1742–1753, 2018.

[8] 石垣達也, 町田和哉, 小林隼人, 高村大也, 奥村学. Distant supervision による質問要約. 情報処理学会第 236 回自然言語処理研究会, 2018.

[9] Felix A. Gers, Jürgen Schmidhuber, and Fred A. Cummins. Learning to forget: Continual prediction with lstm. *Neural Computation*, Vol. 12, pp. 2451–2471, 2000.

[10] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, Vol. abs/1409.0473, , 2014.