文の分散表現を用いた柔軟な質問文検索の試み

福田 りょう　　井佐原 均
豊橋技術科学大学 言語情報学研究室

1 はじめに

ニューラルネットワーク技術の発展と共に, 単語や文章をベクトル化する研究が盛んに行われている. それにより作られた単語や文の実数ベクトルは分散表現と呼ばれ, 計算可能な特徴表現として注目されている. 例えば2文の分散表現のコサイン類似度を測ることで, 文同士の類似度が計算できる.

ところで, サービスを提供する企業の多くが, そのウェブページ上に Frequently Asked Questions(FAQ) を公開している. FAQ は, サービスや製品に関したよくある質問と, その回答を集めたデータである. ユーザーが質問を入力でき, 関連度の高い Q&A を FAQ データから探して表示するという形でよく見られる. FAQ は, ユーザーが知りたい情報を素早く入手する手段として有用であるが, その入力に厳密性が求められるという難点がある. 例えば FAQ 内に存在する「フォルダの削除方法を教えてください」という質問に対する検索クエリを考える. 質問文に含まれる単語を羅列した「フォルダ　削除」や, 質問文の一部と完全一致する「フォルダの削除方法」などで検索が可能である. しかし,「どのようにしてフォルダを削除できますか?」「フォルダが削除できないよ」のような表現が大きく異なる文での検索は難しい.

上に挙げた検索困難な例では, 質問文と検索クエリの文構造が大きく異なるが, 意味的な違いは小さい. 分散表現を用いて文の類似度が計算できると述べたが, 意味的な類似度を測ることができればそれを用い,FAQ において構造が異なるクエリでの検索が可能になると考えた. そこで本研究では, 文の分散表現を用いた柔軟な FAQ システムの構築を目指し, 柔軟な質問文検索法の研究を行った.

本研究により, さまざまな構造のクエリによる検索が可能になった. 例えば,「動画をページ上に掲載したいのですが容量に制限はありますか.」という質問について,「動画の容量に制限はあるか.」や「動画ファイル　容量制限」というクエリでの検索を可能にした.

2 関連研究

分散表現はその獲得法により二つに大別される [1]. 一つは, 共起頻度に基づく古典的な共起頻度ベクトル（カウントベースの分散表現）で, もう一つはニューラルネットワークによる学習から獲得する埋め込みベクトル（ニューラルベースの分散表現）である. 一般に, ニューラルネットワークの発展に伴い近年台頭してきたニューラルベースの分散表現は, カウントベースの分散表現よりも優れている [2].

ニューラルネットワークによる分散表現の獲得手法に Paragraph Vector[3] がある. 教師なし学習による優れた文章の分散表現獲得手法の一つであり, 文書分類タスクなどに適応できる. また Paragraph Vector は通称 Doc2Vec と呼ばれるが, これは Word2Vec[4] に由来する. Word2Vec は「単語の意味はその単語が出現した際の周辺単語によって決まる」という分布仮説に基づいた単語の分散表現獲得手法である.Paragraph Vector は Word2Vec を文章へ拡張したものだと言える. 文の分散表現獲得手法には他に,Skip-Thought[5] というものがある. これは, 対象文に対しその前後の文を予測するニューラルネットワークモデルからなる.

Paragraph Vector と Skip-Thought はいずれも, 教師なし学習によるニューラルベースの手法である. 本研究では, 分散表現獲得のためにこの 2 手法を用いた.

3 提案手法

システムの概要を図 1,2 に示す.

図 1 はシステムの概要図である.FAQ データの中から検索クエリに類似した質問文を探し, 対応する回答を返すのがおおまかな流れである. 類似文の選択には分散表現を用いる. 図 2 のようにクエリと, 全ての質問文との類似度を測る. 類似度で順位付けし上位トップ n を表示するのが, 目指すシステムの動作である.

このシステム作成にあたり, より正しく文の類似度をとれることが重要である.

そこで本研究では分散表現の生成に焦点を当て, 正しい質問が選択できる分散表現の獲得手法を提案する.

Copyright(C) 2019 The Association for Natural Language Processing.
All Rights Reserved.

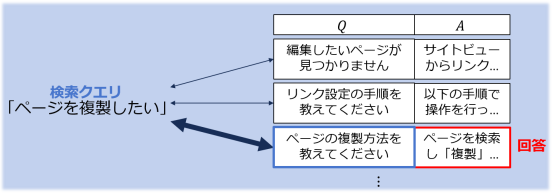

図 1: 概要

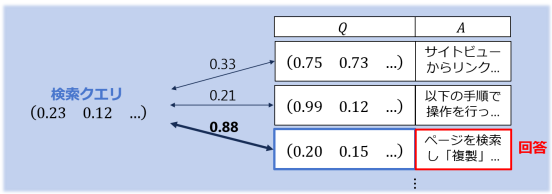

図 2: 内部構造

3.1 名詞の抽出

言語をニューラルネットワークモデルに通す際, 事前にトークナイズする必要がある. 日本語の場合は形態素解析を行う. 本研究では形態素解析に MeCab[6] というツールを用いる.MeCab で形態素解析を行うとトークンに品詞情報が付与される.

4.6 で後述するが, 全てのトークンを用いて分散表現の生成を行うとその検索精度は低い. これは文意を量るのに不要な品詞が含まれているためである. そこで, 不要な品詞を取り除くことを考えた. 実験の結果, 名詞のみを用いることで意味的類似度をとるのに適した分散表現が生成できることが明らかになった.

ここでは品詞情報を基に, 名詞以外の品詞を取り除くことを提案する. 名詞のみ抽出することで,「日本の首都はどこですか?」「日本の首都ってどこ」「日本 首都　どこ」これらは全て [日本, 首都, どこ] のようにトークナイズされる. これにより, 敬体と常体, 文と単語の羅列, のようなクエリの構造の違いを無視できるため柔軟な検索に対応できる.

3.2 2 ベクトル化

3.1 によって向上が見られたが, まだシステムとして十分な精度とは言えなかった. その原因は名詞の持つ特性を考慮していないためであった. 名詞は, 低頻度語ほど文の特徴として有用である. 文中のある名詞がその文にのみ登場するならば, 名詞はその文固有であるからだ. だからと言って, 高頻度が不要な訳では無い. 高頻度語も文を特定するのに必要な情報であることが多い. 特に, 高頻度に含まれる「どこ」「いつ」「何故」のような単語は, 文の種類を大別するのに有用である.

高頻度語, 低頻度語共に重要な情報であるが, ベクトル化の際互いが混じることは望ましくない. そこで,3.1 によって抽出した名詞を, 更にコーパス中の出現回数によって 2 分割し, それぞれからベクトルを作ることを提案する. これは高頻度語からなるベクトルは「質問の種類」を表し, 低頻度語からなるベクトルは「質問の内容」を表すという仮定に基づく. 最後に 2 つのベクトルを連結することで文の分散表現を生成する.

なおこの操作は学習時ではなくテスト時にのみ行うことに留意する.

4 実験

4.1 タスク

検索クエリと, テストコーパスの分散表現を作成し, 類似度を計算してその値を評価する.

4.2 学習コーパス

モデルは, ウェブ小説投稿サイト「小説家になろう」の小説 100 作品を用いた. これは全体で 35.6MB, 約 33 万文の日本語テキストデータである.

尚, 大規模な日本語データとして Wikipedia の日本語記事を使うことを検討したが, 話し言葉で書かれたテストコーパスに対し書き言葉である Wikipedia を用いることは不適切であるとし却下した. しかし, 名詞のみを使って構造の違いを削除したことでこの問題は無くなり, 実際に Wikipedia も用いたが, 小説コーパスの方がより優れた結果を残した. これについては説明がつく理由に思い至らなかった.

4.3 テストコーパス

協力企業にいただいた 885 組の Q&A データからなる FAQ の中から,Q（質問）をテストコーパスとして用いた.

4.4 検索クエリ

テストコーパスからランダムに選んだ質問を, ユーザーが質問しそうな形に言い換えたものを 50 文用意した.

4.5 評価方法

検索クエリに対し全ての質問文との類似度を測り, 順位付けする. 正しい質問文とクエリの類似度の順位が高いほど良い. この順位を今後類似順位と呼ぶ.

50 個の検索クエリの類似順位について, その平均値, 中央値, 1 位・5 位以内を取った個数（Top1,Top5）で評価を行う.

Copyright(C) 2019 The Association for Natural Language Processing.
All Rights Reserved.

4.6 ParagraphVector と Skip-Thought

2 で記述した二つの分散表現獲得手法を比較する. 以下に学習条件を示す.

表 1: 学習条件

手法	ベクトルサイズ	エポック数	単語ベクトルサイズ	最低登場回数
Paragraph Vector	200	100	-	10
Skip-Thought	200	100	100	-

結果は以下である. ここでは手法 3.1,3.2 を用いないことを明記しておく.

表 2: Paragraph Vector と Skip-Thought

	Paragraph Vector	Skip-Thought
平均値	278.6	218.4
中央値	111.5	97.5
Top1	4	3
Top5	10	5

平均値・中央値からは Skip-Thought がやや高い順位を取れていることが読み取れるが, Top1・5 は Paragraph Vector の方が良い数字である.

4.7 名詞の抽出

ここでは, 上で比較した 2 手法に対し 3.1 名詞の抽出を適用し, その効果を検証する.

表 3: 名詞抽出

	Paragraph Vector	Skip-Thought
平均値	71.46	126.34
中央値	2	55.5
Top1	22	9
Top5	32	12

両手法共に精度を向上させたが,Paragraph Vector の改善が顕著である.

Paragraph Vector（P-V）に名詞の抽出を適用することで類似順位を上げた一例を示す. 表中の数値は類似順位を表している.

表 4: 改善例:名詞抽出

検索クエリ	動画ファイルの容量に制限はありますか.
質問文	動画ファイルを直接ページ上に掲載したいのですが, 容量に制限はありますか.
P-V	856
P-V(名詞抽出)	1

表 4 の質問文は, 形態素解析の結果 22 個の形態素にトークナイズされる. そのうち助詞や動詞, 記号など名詞以外の形態素は 13 個と多い. これらを文の意味と無関係とし削除することで, 類似順位を 856 位から 1 位にまで向上させた.

Paragraph Vector に名詞の抽出を適用することで, 大きく検索精度が向上することが分かった. 今後の実験は Paragraph Vector と名詞抽出を前提とする.

4.8 高頻度語の削除

3.2 で, 名詞は, 低頻度語ほど文の特徴として有用であるとした. これを確かめるために, 高頻度語を一律に削除して実験を行う.

4.9 閾値の決定

削除基準となる閾値を決定するため, テストコーパスの質問 885 文内の名詞を全て数え上げ, 語彙数や単語の登場回数を調べた. コーパス内の語彙数は 588 語と小さく, 全単語の出現回数の総和である総登場回数は 6574 回であった. その中で, 登場回数が 100 を超える単語は 13 個あり, それらの登場回数の合計は 2226 回となっている. これは全体の 1/3 以上を占めており, さらに 449 単語は登場回数 10 回未満と, 高頻度語と低頻度語がはっきり分かれたコーパスであることが分かった. そこで 100 回以上出現の単語を高頻度語とし, これを満たす 13 単語を削除してタスクに適用しその精度を見る.

表 5: 高頻度語削除の検証

	P-V	P-V(高頻度語の削除)
平均値	71.46	128.24
中央値	2	3.5
Top1	22	16
Top5	32	30

高頻度語を削除することにより, 検索精度は下がってしまった. 一例を以下に示す.

表 6: 改悪例:名詞抽出

検索クエリ	ファイルリンク　設定
質問文	ファイルリンク設定の手順を教えてください.
P-V	2
P-V(高頻度語削除)	492

「ファイル」「リンク」「設定」全て高頻度語である. 全てのトークンが削除されて意味のない分散表現になってしまった.

4.10 2 ベクトル化の導入

4.8 では, 高頻度語の削除により検索精度を下げた. 情報を落とさず高頻度語と低頻度語を区別する方法として, 提案手法 3.2 を実装して評価する.

表 7: 2 ベクトル化の検証

	P-V	P-V(2 ベクトル化)
平均値	71.46	38.53
中央値	2	2
Top1	22	24
Top5	32	35

正しく精度が向上した. 一例を以下に示す.

Copyright(C) 2019 The Association for Natural Language Processing.
All Rights Reserved.

表 8: 改善例:2 ベクトル化

検索クエリ	ファイルリンク　設定
質問文	ファイルリンク設定の手順を教えてください.
P-V(高頻度語削除)	492
P-V(2 ベクトル化)	1

4.8 で名詞削除により生じた問題を解決した他, 平均順位の上昇から全体的な検索精度が向上したと言える.

5　考察

実験で, 学習モデルに Paragraph Vector を用い, 提案 3.1,3.2 を適用することで精度の高い検索ができることを示した. 次に類似順位が高い例を表 9 に, 低い例を表 10 に示す.

表 9: 正しい検索例

	検索クエリ	質問文	類似順位
単語の羅列	ページ　更新回数	各ページの更新回数を確認することは可能ですか.	1
常体	行間を編集する方法を教えてくれ	テキストの行間が異なっています. 編集方法を教えてください.	1
敬体	セルを縦に繋げるにはどうすればいいですか.	セルの縦結合を行うことはできますか.	1

単語の羅列, 敬体の文, 常体の文など様々な形のクエリで柔軟な検索が行えるようになった.

表 10: 正しくない検索例

	検索クエリ	質問文	類似順位
文長の異なり	ページプロパティの変更はそのページだけ反映されるのですか.	ページプロパティから所属課室の変更を行いました. 変更内容が反映されるのは, 変更を行ったページのみという認識でよろしいでしょうか.	79
単語の言い換え	ページ　コピー	作成を行ったページの複製を行うことは可能ですか.	126

検索クエリと質問文との文長の違いや,「複製」から「コピー」へのような単語の言い換えにより, 類似順位が下がってしまう. これらの問題はいずれも, 今回実験を行った全ての手法に共通している.

6　おわりに

本研究では正確な入力を必要とする FAQ システムにおいて, 分散表現を用いることで柔軟な質問文検索を可能にすることを試みた. 分散表現獲得モデルをそのまま用いるだけでは実用的な精度に至らなかったが, 名詞の抽出と 2 ベクトル化により大きく精度を上げ, 柔軟な検索を可能にした. 一方で, 文長が大きく異なると精度が低下する, 同じ意味の単語の言い換えに対応できないといった課題も残されている. また, エポック数やベクトルサイズ, 最低登場回数など Paragraph Vector モデルのパラメータや, 高頻度語と低頻度語の閾値など, より精度を高めるための最適な値の検討が不足していると感じた. 今後は, 残された課題に取り組むと共に, これらパラメータについても最適な設定を検討していきたい.

参考文献

[1] 鷲尾 光樹. 語の分散表現と上位下位関係-研究動向と今後への提案-. 人工知能学会　インタラクティブ情報アクセスと可視化マイニング研究会 (第 13 回) SIG-AM-13-03, 2016.

[2] Marco Baroni, Georgiana Dinu, and Germán Kruszewski. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 238–247. Association for Computational Linguistics, 2014.

[3] Quoc V. Le and Tomas Mikolov. Distributed representations of sentences and documents. In *ICML*, volume 32 of *JMLR Workshop and Conference Proceedings*, pages 1188–1196. JMLR, 2014.

[4] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. volume abs/1301.3781, 2013.

[5] Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Skip-thought vectors. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 3294–3302. Curran Associates, Inc., 2015.

[6] Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. Applying conditional random fields to japanese morphological analysis. In *Proceedings of EMNLP*, pages 230–237, 2004.

Copyright(C) 2019 The Association for Natural Language Processing.
All Rights Reserved.