

文のマルチカテゴリ分散表現の獲得とその応用

谷 健太郎[†] 上野 史[‡] 太田 学[‡]

^{† ‡}岡山大学大学院自然科学研究科 〒700-8530 岡山県岡山市北区津島中三丁目1番1号

E-mail: [†]plpa5yw5@s.okayama-u.ac.jp, [‡]{uwano, ohta}@okayama-u.ac.jp

あらまし 言語処理を行うニューラルネットワークを用いたモデルでは、入力文と正解アノテーションからモデルを学習して予測を行うことが多い。本稿では入力文に付随する情報を活用した分散表現を獲得するために、入力文に付随する複数の情報から属性を定義し、その属性カテゴリごとに分散表現を生成するマルチカテゴリ分散表現生成モデルを2つ提案する。実験では、属性を持つ文を作成し、それを使った類似文予測を行う。また含意関係認識にも対して提案モデルを応用する。

キーワード 機械学習、自然言語処理、ニューラルネットワーク

1 はじめに

近年、単語や文の分散表現を活用した研究が活発に行われている。例えば、単語やトークンの分散表現(トークンベクトル)を生成するモデルにGloVe-BoW[1]やBERT(Bidirectional Encoder Representations from Transformers)[2]などがあり、文単位の分散表現(文ベクトル)を生成するモデルにInferSent[3]、SBERT(Sentence-BERT)[4]などがある。また、文や文集合から必要な情報の抽出や関係の予測などを行う際に、文ベクトルを用いると単語やトークンの分散表現を用いるよりも効率よく抽出や予測をすることができる。ここで、文ベクトルには目的タスクで必要となる情報がマッピングされていなければならぬが、特に学習データが少ない場合や偏りがある場合は過学習が起こり、マッピングされる情報に偏りや抜けが起こる。少ないデータでも過学習を起こさず、文から必要な情報を分散表現にマッピングする方法のひとつに、事前学習により文ベクトルを生成するモデルを用いて、目的タスクのために微調整(Fine-tuning)する方法がある。しかし事前学習で文ベクトルにマッピングした情報が不十分なとき、多くのFine-tuningが必要になる。また、学習データが少ないとFine-tuningが難しくなるため、事前学習での目的タスクにも有用な情報をマッピングできるとよい。

本稿では、モデルの出力する分散表現により多くの情報をマッピングし、性能を向上させたモデルと学習法を提案する。具体的には、入力文の持つ複数の属性を用いて学習する文ベクトル生成モデルをBERTを用いて構築し、そのモデルと学習法を提案する。

また、複数属性を持つ文のデータセットは、評価対象と評価内容の情報が付随するレビュー文を用いて作成する。具体的には、楽天商品レビューデータセット¹を用いて、属性カテゴリ、属性内容を定義し類似文予測(Semantic Textual Similarity)のデータセットを作成する。さらに、NTCIR-10 RITE-2 デー

タセット²を用いた含意関係認識(Recognizing Inference in Text)タスクの評価実験を行った。

2 関連研究

様々なタスクで分散表現を利用したモデルが研究されており、トークンや単語の分散表現を生成するモデルには文脈を考慮しないGloVe-BoWや文脈を考慮するBERTなどがある。文や文書の分散表現を生成するモデルはトークンの分散表現の集約、統合等を行うことで文ベクトルを生成するものが多く、InferSent、SBERTなどがありモデルの学習と合わせて提案されている。

2.1 BERT

Multi-Head Attentionを基本としたTransformer[5]Layerを数層積むことで構成されているBERTやBERTを基にしたモデルは事前学習を利用して、様々な言語処理タスクで高い性能を発揮する。これらのモデルは入力文をトークンごとに分割し、トークンに対応する分散表現(トークンベクトル)を生成して様々なタスクに利用される。また、入力文全体の性質などは基本的に入力トークンに追加するCLSトークンの出力ベクトルを利用し予測される。BERTの事前学習タスクは、区切られた入力文が連続したものかを予測するNSP(Next Sentence Prediction)、マスク部分のトークンを予測するMLM(Masked Language Model)があり、BERTの軽量化したモデルALBERT[6]ではNSPの負例に元の文のトークン列を逆順にしたものを使い正しい順序か予測するSOP(Sentence Order Prediction)が用いられている。

2.2 SBERT

Sentence-BERTは、BERTの出力であるトークンベクトルをMEAN/MAX/CLSの3手法でまとめ、文ベクトルを生成し比較している。学習はBERTの事前学習に加え、SNLI[7]と

¹: 楽天株式会社 (2020): 楽天市場データ. 国立情報学研究所情報学研究データリポジトリ. (データセット). <https://doi.org/10.32130/idr.2.1>

²: NTCIR Project NTCIR-10 RITE. <http://research.nii.ac.jp/ntcir/permission/ntcir-10/perm-ja-RITE.html>

表 1: トーケンベクトル生成 (Encoding)

手法	説明
Embedding	入力トーケンと一対一に対応するベクトルを生成
CNN	CNN による位置的に近いベクトルの情報を加味したベクトルの生成
RNN	RNN で系列を単方向（もしくは双方向）から処理各処理と前の処理の情報を加味したベクトルの生成
Attention	Attention 構造による入力全体の情報を加味したベクトルの生成

各手法は Embedding + α の処理

Multi NLI [8] のあわせて約 100 万ペアの入力文の含意関係を矛盾 (contradiction), 含意 (entailment), 中立 (neutral) から予測する。NSP では連続した文の単語のトピックが似通ったものになる可能性が高いことにより、文脈的な意味を考慮せずに単語のトピックによる予測でも性能が見込めるところから、文脈を学習しない可能性がある。それを防ぐために SOP が用いられる。

3 分散表現生成モデルの段階的な構成

言語処理を行うニューラルネットワークモデルの多くはトーケンベクトルや文ベクトルを生成し、文ベクトルを基にタスクに必要な形の出力を生成しているが、モデルには様々なものがある。本稿ではトーケンベクトルや文ベクトルを生成するモデルの構成を Tokenization, Encoding, Pooling, Concatenation の 4 つに分け、以下に説明する。

まず、モデルに入力した文、トーケンに分割する Tokenization について述べる。トーケンには、最小の文字から出現頻度を用いたサブワード、形態素解析による文法構造を用いた単語などがある。BERT の学習済み公開モデルでは、用いられる語彙に合わせて、サブワードによるトーケン化が多く用いられる。サブワードとは、文法的な分割を行う単語とは異なり、文を分割した文字列の出現頻度から分割する長さを学習して決定する語彙のことである。

次に、Encoding について述べる。Encoding はトーケンに対応した分散表現を生成し、出力する。Encoding の主なモデルを表 1 にまとめる。Embedding により語彙をベクトル化した後に、他のトーケンベクトルの情報を加味した処理を行うほど、複雑なタスクに対応できるモデルとなる。それぞれ生成する次元の大きさやウィンドウ幅、フィルタ数、重ねる層の数、他の処理との組み合わせなどを変えることでより高度なタスクに対応できる複雑なモデルになる。ただし、モデルが複雑になるほど学習が難しくなり過学習などの問題が起こりやすくなる。また、BERT [2] は多層的な Attention の発展した構造といえる。その後、Encoding で得た複数のベクトルから必要な情報を集約し、統合するのが Pooling である。Pooling の集約・統合方法を表 2 にまとめる。Pooling によって、トーケン毎の分散表現をまとめ、入力文全体の情報を集約した一定数の文ベクトルを得ることができる。

表 2: 複数ベクトルからの集約・統合 (Pooling)

手法	統合結果のベクトル
CLS	先頭 CLS トーケンのベクトル
MAX	要素ごとに最大値を集めたベクトル
MEAN	要素ごとに平均値を集めたベクトル
Attention	Attention 構造でまとめたベクトル

表 3: 2 つのベクトルの統合 (Concatenation)

手法	説明
(\mathbf{x}, \mathbf{y})	元のベクトルを並べて連結後に全結合層で次元を調整
$(\mathbf{x} - \mathbf{y})$	要素差の絶対値のベクトル
$(\mathbf{x} * \mathbf{y})$	要素積のベクトル
$(\mathbf{x} * \mathbf{y}, \mathbf{x} - \mathbf{y})$	要素差の絶対値と積のベクトルを連結後に全結合層で次元を調整
$(\mathbf{x}, \mathbf{y}, \mathbf{x} - \mathbf{y})$	元のベクトルと要素差の絶対値のベクトルを連結後に全結合層で次元を調整
$(\mathbf{x}, \mathbf{y}, \mathbf{x} * \mathbf{y})$	元のベクトルと積のベクトルを連結後に全結合層で次元を調整
$(\mathbf{x}, \mathbf{y}, \mathbf{x} - \mathbf{y} , \mathbf{x} * \mathbf{y})$	元のベクトルと要素差の絶対値と積のベクトルを連結した後に全結合層で次元を調整

最後に、2 つのベクトルが持つ情報を加味したベクトルを生成する Concatenation について述べる。その方法を表 3 にまとめる。Concatenation では、文を比較したり関係を予測したりするとき、2 つの入力文の文ベクトルから、その関係性を表すベクトルや 2 つのベクトルが持つ情報を足し合わせた情報を持つベクトルを生成する。2 文の類似文予測や含意関係認識では、文ベクトル間に Concatenation を用いて関係を表すラベルを予測するためのベクトルを生成する。また、SBERT [4] では含意関係認識を行う際に $(\mathbf{x}, \mathbf{y}, |\mathbf{x} - \mathbf{y}|)$ と $(\mathbf{x}, \mathbf{y}, |\mathbf{x} - \mathbf{y}|, \mathbf{x} * \mathbf{y})$ の性能が高かったことが報告されている。

4 分散表現のマルチカテゴリ化

文の性質を予測する際には文ベクトルを生成し、それを用いて予測するが、タスクや着目点など注目したい事柄を変えたとき、異なる性質を持っている場合がある。例えば、観点付き感情極性推定では注目する観点を変えたとき感情極性が異なる場合は多い。他にも複数の事柄について述べた文とその内のひとつの事柄について同じ内容を述べた文は、ある事柄については同じ情報を持っているため、文ベクトルとしては類似すべきである。しかし、一方の文は複数の事柄についても述べているため、両ベクトルは単純には類似しない。これは、比較する 2 文に対して 3 章で述べた Concatenation を実装したモデルを対象とした事柄について学習させるか、文ベクトル生成モデル全体を再度学習することで解決できる。しかし、複数の事柄について同時に学習することで、より多くの情報を中間表現にマッピングし、かつ必要に応じて対象の事柄に対する文ベクトルを生成できれば、中間表現にマッピングされたより多くの情報を基にした分散表現を用いた予測で性能が向上することが期待される。

表 4: 入力文の属性例

属性カテゴリ	属性内容
種類	文の種類 (感想/事実/批評/要望/推薦...)
発信元	名前や所属など
発信先	誰に対しての文か
対象	何に対しての文か
視点 A	視点 A における主張など
視点 B	視点 B に対する主張など
.....

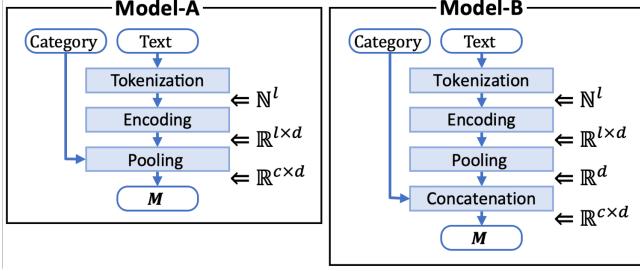


図 1: Model-A と Model-B の構成概要

待できる。そこで、着目点やタスクなどの事柄ごとに文ベクトルを生成し、それらの複数の情報を並行して学習する“文のマルチカテゴリ分散表現生成モデル”を提案する、

5 文のマルチカテゴリ分散表現生成モデル

はじめに文のマルチカテゴリ分散表現は、入力文に付随する情報や注目すべき点等を入力文の属性として与え、その属性のカテゴリごとに生成した文ベクトルの集合と定義する。例えば属性には表 4 のように様々なものが考えられる。ここで属性内容はそれぞれ属性カテゴリにおける入力文の持つ属性を表す。

次に文のマルチカテゴリ分散表現生成モデルの実装について 3 節で述べた Tokenization, Encoding, Pooling, Concatenation の構成を用いて、マルチカテゴリ化を行う異なる 2 つモデルを考える。それらは図 1 に示すように、Pooling においてマルチカテゴリ化する Model-A と Concatenation でマルチカテゴリ化する Model-B である。図 1 では、入力文を “Text”，属性カテゴリを “Category” として、モデルの出力であるマルチカテゴリ分散表現 M が生成される。また、入力文のトークン長を l 、属性カテゴリの数を c 、各分散表現の次元を d と表記している。

次に Model-A, Model-B の実装をそれぞれ図 2, 図 3 に示す。図 2, 図 3 中の記号は式 (1) の通りである。

$$\begin{aligned}
 \mathbf{x} &:= [x_1, \dots, x_l] \in \mathbb{N}^l \\
 \mathbf{A} &:= [\mathbf{a}_1, \dots, \mathbf{a}_c] \in \mathbb{R}^{c \times d} \\
 \mathbf{U} &:= [\mathbf{u}_1, \dots, \mathbf{u}_l] \in \mathbb{R}^{l \times d} \\
 \mathbf{M} &:= [\mathbf{m}_1, \dots, \mathbf{m}_c] \in \mathbb{R}^{c \times d}
 \end{aligned} \tag{1}$$

入力文 “Text” を Tokenization でトークンに分割した最大長 l の id 列 $\mathbf{x} \in \mathbb{N}^l$ を BERT で d 次元のトークンベクトル列

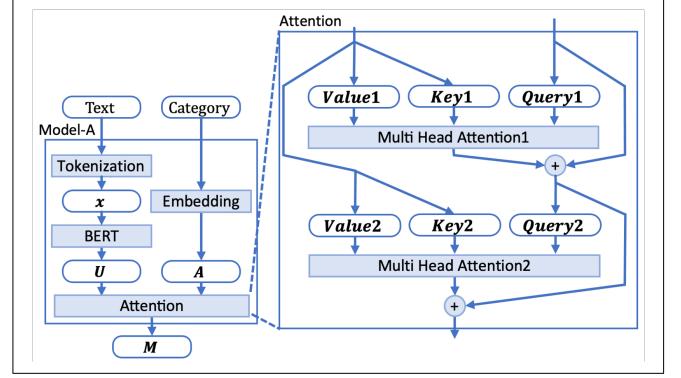


図 2: Model-A の実装

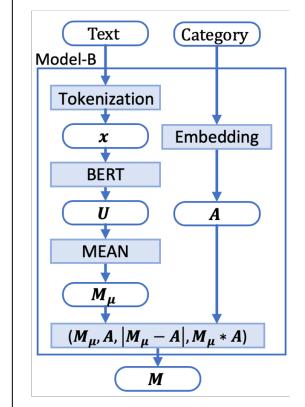


図 3: Model-B の実装

$\mathbf{U} \in \mathbb{R}^{l \times d}$ に Encoding する。

Model-A は、BERT のトークンベクトル \mathbf{U} から文ベクトルを生成する Pooling で入力文に付随する c 種の属性カテゴリ(図中 “Category”)毎に Embedding し、ベクトル化した \mathbf{A} を Query, \mathbf{U} を Key と Value に利用した Attention へ入力する。その Attention の出力を文のマルチカテゴリ分散表現 \mathbf{M} として生成する。また、Attention は Multi-Head Attention を 2 層重ね、1 層目で属性カテゴリに基づく情報を抽出し、2 層目でさらに 1 層目で抽出した情報を加味した情報を抽出できるように実装した。

Model-B では、 \mathbf{U} から Pooling の MEAN で式 (2) のように、平均ベクトルを生成し、次元を \mathbf{A} を揃えるため tile 関数で c 個に複製することで、 $\mathbf{M}_\mu \in \mathbb{R}^{c \times d}$ とする。その後、 \mathbf{A} と属性カテゴリごとに Concatenation の $(\mathbf{x}, \mathbf{y}, |\mathbf{x} - \mathbf{y}|, \mathbf{x} * \mathbf{y})$ を用いて、属性カテゴリごとの文ベクトルへ変換する。

$$\mathbf{M}_\mu = \text{tile} \left(\frac{1}{l} \sum_{i=1}^l \mathbf{u}_i, c \right), \quad \mathbf{M}_\mu \in \mathbb{R}^{c \times d} \tag{2}$$

6 モデルの損失関数

モデルの学習は事前学習 (Pre-training) と微調整 (Fine-tuning) に分けられるが、本稿では Fine-tuning 時に目的タスクに対する損失に加えて、同時に別のタスクにおける損失を用いる。その別のタスクをサブトレーニングタスクと呼び、Pre-training に用いられるマスク補完 (MLM: Masked Language

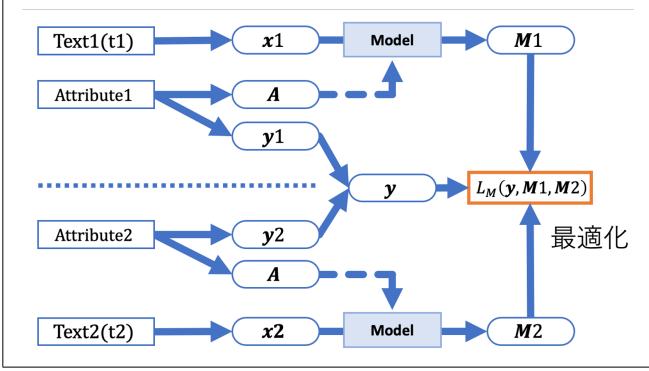


図 4: 損失関数イメージ図

Model), 順序予測 (SOP: Sentence Order Prediction) を採用し, さらに属性を用いた損失を定義して実験に利用する.

損失関数はモデルの出力する分散表現を全結合層などを用いてタスクの分類数や次元に変換し, 目的タスクの正解となるデータと比較し, 計算する. まず, 分類損失関数には式(3)のクロスエントロピーを用いる. 分類数 n の正解ラベルを $\mathbf{y} \in \{0, 1\}^n$, モデル出力を softmax 関数で 0 から 1 に変換したラベルの予測値を $\hat{\mathbf{y}} \in \mathbb{R}^n$ とし, 式(3)のように損失を計算する.

$$\text{CrossEntropy}(\mathbf{y}, \hat{\mathbf{y}}) = - \sum_{i=1}^n y_i \cdot \log(\hat{y}_i) \quad (3)$$

また, 順序予測 (SOP: Sentence Order Prediction) は True/False の 2 値分類, マスク補完 (MLM) は語彙への分類問題として損失を定義する.

次に, 属性を用いた損失関数を SBERT [4] で用いられる Siamese Networks を参考に定義する. その損失関数を属性損失関数と呼び, 対比損失関数を複数属性へ対応させた式(4)の L_M を属性損失関数として定義し, 評価実験で行う属性が付随する類似文予測で用いる.

$$L_M = \sum_{i=1}^c [y_i \cdot (\|\mathbf{m1}_i - \mathbf{m2}_i\| - \alpha) + \varepsilon]_+ \\ y_i := \begin{cases} 1 & (\text{属性 } \mathbf{a}_i \text{ の内容が入力 } t1, t2 \text{ で一致}) \\ -1 & (\text{不一致 or 一方に属性がない}) \\ 0 & (\text{両方に属性 } \mathbf{a}_i \text{ がない}) \end{cases} \quad (4)$$

$$[z]_+ := \max(z, 0)$$

式(4)は図4のように, 文ペア $t1, t2$ を入力するときに用いる損失関数である. 入力文 $t1, t2$ を文ベクトル生成モデル (図中の Model) に入力した際の出力されるマルチカテゴリの文ベクトル列を $\mathbf{M1}, \mathbf{M2}$, 入力文 $t1, t2$ どちらかが持つ属性のカテゴリ c 種のうちの任意のカテゴリ i に対応する属性カテゴリベクトルを \mathbf{a}_i , 文ベクトルを $\mathbf{m1}_i, \mathbf{m2}_i$ とする. また, 入力文 $t1, t2$ で属性の内容が一致しているかを表す値として i に対応する属性内容が一致しているとき $y_i = 1$, 不一致のとき $y_i = -1$, 入力文に属性がないときを $y_i = 0$ とする $y_i \in \{-1, 0, 1\}$ を定義する. ここで, $\alpha \in \mathbb{R}$ は距離の閾値となる定数で, $\varepsilon \in \mathbb{R}$ は同属性と異属性での距離の閾値 α のマージンとして利用する定数である. L_M の意味としては, 全ての属性カテゴリにおいて

入力文 $t1, t2$ の文ベクトル間の距離を属性の内容が一致しているときは $\alpha - \varepsilon$ 以下, 不一致のとき $\alpha + \varepsilon$ 以上になるよう学習する損失関数である.

7 評価実験

Model-A, Model-B の評価実験として, 次のタスクを通して比較する.

(1) 類似文予測

2つのレビュー文が類似文であるか否かを予測する. 正しく予測した割合 (Accuracy) で評価する.

(2) 含意関係予測

2つの文の含意関係を予測する. 含意関係を表すラベルを正しく予測した割合 (Accuracy) で評価する.

類似文予測では, 文に追加して付隨する情報を用いて属性を定義し, その属性を用いて学習を行うことでモデルの性能が向上するかを比較する. 含意関係認識では, 属性を用いない分類問題を解くモデルとして比較する. それぞれ学習では, バッチサイズは 256, 学習回数 (step 数) は検証データにおける損失 (val_loss) の値が更新されないか悪化したかを判定し自動で終了する. 停止条件は, 最新 3 回分の検証データによる予測において損失の平均がさらに前の 3 回の平均以上になる回数が, それまでの損失の最小値を更新しないまま 6 回以上になったタイミングとする. 検証の頻度はデータ 5000 件毎か 1 epoch 毎とする. また 5 節で述べた Model-A, Model-B, 学習にサブトレーニングタスクにマスク補完 (MLM) と順序予測 (SOP) を用いるモデルを Model-A.SUB, Model-B.SUB, さらにサブトレーニングタスクに式(4)の損失関数 L_M を用いたモデルを Model-A.SUB_attr, Model-B.SUB_attr とする. ただし, 属性損失 L_M は属性を複数定義している楽天商品レビューを用いた類似文予測でのみ用いる. また, Encoding に単純な Embedding を用いたものを line_attn, BiLSTM を用いたものを bilstm_attn とし, それらの Pooling では, Target を属性カテゴリベクトル \mathbf{A} とする Source-Target Attention を用いる. Source-Target Attention は, Multi-Head Attention の Head 数とレイヤ数を 1 としたもので, 単純な構成のモデルとして実験する. また, BERT は日本語 Wikipedia を対象に情報通信研究機構データ駆動知能システム研究センターで事前学習を行った BERT モデル³ を利用し, 学習に利用する.

7.1 類似文予測実験

類似文予測 (STS: Semantic Textual Similarity) は, 文章の類似性を判別するタスクで類似するか否か (STS_True, STS_False) の二値分類タスクとして実験する. 学習には, 属性損失を用いるモデルでは類似文か否かに加えて, 入力文が持つ属性を利用する.

モデルの学習および評価に使うデータセットとして, 楽天商

³: NICT BERT 日本語 Pre-trained モデル <https://alaginrc.nict.go.jp/nict-bert/index.html>

表 5: 楽天商品レビューの入力文の属性

属性カテゴリ	内容
レビュー評価	高評価/低評価
店舗 ID	固有の ID
商品 ID	固有の ID
商品ジャンル	商品のジャンル x
先祖ジャンル	ジャンル a, ジャンル b, ...
使い道	実用品・普段使い/ビジネス/イベント/ おもたせ・ギフト/プレゼント/趣味/None
購入目的	男性(彼, 夫)へ/子供へ/仕事関係へ/友人へ/ 女性(彼女, 妻)へ/家族へ/親戚へ/自分用/None
頻度	はじめて/リピート/None

品レビューデータセットを用いて類似文ペアのデータセットを作成する。データセットでは、レビュー集合から条件付きで抽出した複数のレビューから 1 つのレビューとそのレビューの正例ペアと負例ペアで計 3 レビューをまとめて 1 件のデータとする。データセットの正例、負例は、レビューデータのレビュー本文に対して付随する対象となる商品への評価、対象となる商品の情報を用いて定義する。対象商品の情報には商品 ID や商品ジャンル、商品説明文などがある。また、各レビューに対してレビューの持つ情報から属性を定義する。その属性のカテゴリ及び内容を表 5 に示す。レビュー評価はレビューに付随する評価ポイントと呼ばれる項目が {0,1,2} 点のものを低評価、{4,5} 点を高評価とし、店舗 ID、商品 ID はそれぞれ店舗と商品に付与されている固有の ID。商品ジャンルは商品の種類を階層的に分類したもので、例えば商品ジャンルが「カジュアルシャツ」のとき「メンズファッション/トップス/カジュアルシャツ」というように先祖となるジャンルが定義されており、商品毎にジャンルとその先祖ジャンルがある。属性には先祖ジャンルの上位の第一位から第三位までを用いる。使い道、購入目的、頻度はそれぞれの内容が文字列で定義されている。これらの属性が入力文の各ペアで一致するか否かを式 (4) の y_i のように定義する。

次に、入力文ペア生成用にレビュー全体から一定数のレビューを抽出する。実験では一度の抽出数を 200 レビューとし、正例を作成しやすくするため、商品ジャンルの第一位を用いて抽出する。モデルの学習には 10 万件、検証データは同条件で異なるレビューから 2 万件のデータを作成し、テスト用データでは学習データと異なるジャンルのレビューから 2 万件のデータを作成する。

最後に抽出したレビューの組み合わせから、正例ペアと負例ペアを持つデータを生成する。正例ペアは属性の商品ジャンルとレビュー評価が一致し、かつレビュー本文の Trigram matching⁴ のスコアが同じ文を意味する 1.0 以外で最上位のペアを用いる。負例ペアは、ランダムに抽出したレビューを用いる。

4 : PostgreSQL 12.4 文書 付録 F 追加で提供されるモジュール F.31. pg_trgm
<https://www.postgresql.jp/document/12/html/pgtrgm.html>

7.2 含意関係認識実験

含意関係認識 (RITE: Recognizing Inference in TExt) は 2 文の関係性を予測するタスクである。マルチカテゴリ分散表現生成モデルにおける 1 つの属性カテゴリをタスクとして、その分散表現を基に分類し、学習及び予測を行う。実験に用いる含意関係認識のデータセットは NTCIR-10 の RITE-2 を利用し、Binary-Class 分類と Multi-Class 分類の 2 つの実験を行う。それぞれのタスクにおける正解ラベルを以下に説明する。

(1) Binary-Class 分類

入力 2 文のペアの含意関係を「含意する」(ラベル Y) か「含意しない」(ラベル N) の 2 値分類

(2) Multi-Class 分類

入力 2 文のペアの含意関係を「単方向に含意する」(ラベル F) 「双方向に含意する(換言)」(ラベル B) 「矛盾する」(ラベル C) 「それ以外(無関係)」(ラベル I) の 4 値分類

また、学習データからランダムに抽出した 5 分の 1 を検証データとして過学習を検知し、学習を停止するために利用する。

7.3 実験結果

7.3.1 類似文予測実験の結果

類似文予測タスクの評価実験の結果を示す。モデルの学習における損失 (Loss) と正解率 (Accuracy) の推移を図 5 と図 6 に示す。グラフの step はバッチサイズのデータ 256 件の学習を 1 step として実験し、図の縦軸は、学習データの Loss と Accuracy を train_acc, train_loss 過学習を検知して学習を停止するための検証データの Loss と Accuracy を val_acc, val_loss と表記する。また、学習を終了したモデルに評価用のテストデータで予測し、各モデルの正解率 (Accuracy) をそれぞれ表 6 にまとめる。

まず、line_attn, line_attn_attr, bilstm_attn, bilstm_attn_attr について、それぞれ図 5 における val_loss が僅かに減少し続け学習の停止が遅くなっているが、図 6 の train_acc が他モデルと比べ低く、表 6 の結果からも予測精度は低いことが分かる。よって、BERT と比べ単純な Encoding を用いるこれらのモデルでは、このタスク、データにおいて予測が難しいことが分かる。

次に Model-A, Model-B では、図 5 からは約 1000 step で val_loss が減少しなくなり、学習が停止している。一方、サブトレーニングタスクを用いた Model-A_SUB, Model-A_SUB_attr, Model-B_SUB, Model-B_SUB_attr は約 1500 step で val_loss が減少しなくなり、学習が停止している。これは学習に用いる損失の種類が増えたことで、学習に考慮すべき情報が増えた影響でより多くの学習が必要になったためだと見える。次に表 6 からそれぞれ予測精度を比較すると、Model-Aにおいてはサブトレーニングタスクに MLM, SOP を用いた Model-A_SUB の Accuracy が約 0.5 ポイント向上し、さらに属性損失 L_M を用いた Model-A_SUB_attr は Model-A_SUB と比べ約 0.6 ポイント向上している。一方、Model-B_SUB の

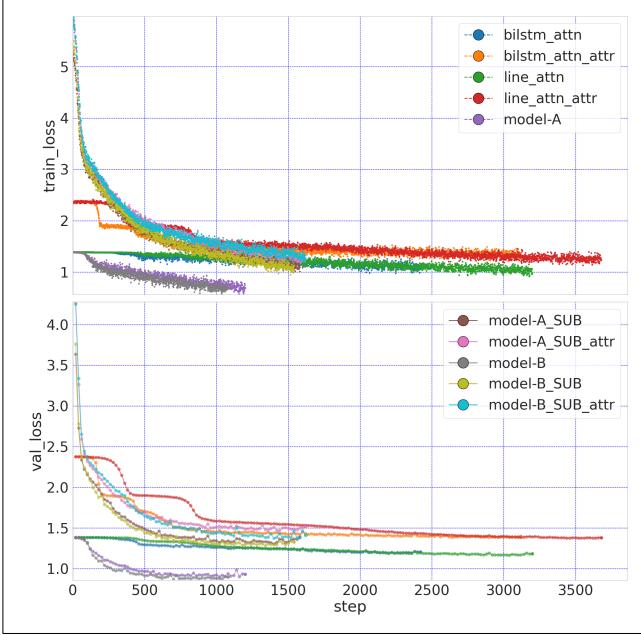


図 5: 類似文予測: Loss の推移

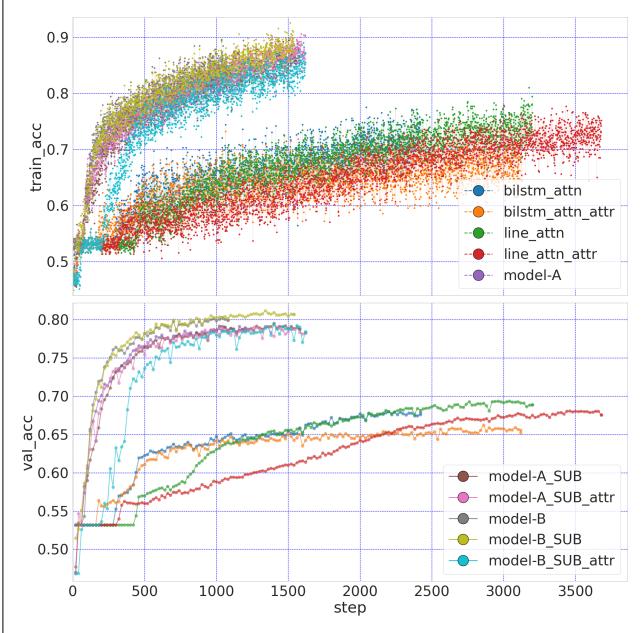


図 6: 類似文予測: Accuracy の推移

Accuracy は Model-B と同程度で、Model-B_SUB_attr は Accuracy が約 1.8 ポイント低下している。よって、この実験において、Pooling 時にマルチカテゴリ化している Model-A は、サブトレーニングタスクに MLM, SOP, 属性損失 L_M を導入することで予測精度を向上させる効果が確認できた。対して、Pooling において MEAN でトークンベクトルの平均ベクトルを生成し、Concatenation でマルチカテゴリ化した Model-B には、どのサブトレーニングタスクも効果は確認できなかった。

7.3.2 含意関係認識実験の結果

含意関係認識タスクにおける実験の結果を示す。損失 (Loss) と正解率 (Accuracy) の推移を Binary-Class 分類は図 7, 図 8, Multi-Class 分類は図 9, 図 10 に示す。また、類似文予測と同

表 6: 類似文予測: テストデータ予測結果

Model	Accuracy
bilstm_attn	0.6381
bilstm_attn_attr	0.6210
line_attn	0.6483
line_attn_attr	0.6305
model-A	0.7131
model-A_SUB	0.7188
model-A_SUB_attr	0.7245
model-B	0.7359
model-B_SUB	0.7355
model-B_SUB_attr	0.7174

様に、グラフの step はバッチサイズのデータ 256 件の学習を 1 step として実験し、図の縦軸は、学習データの Loss と Accuracy を train_acc, train_loss, 過学習を検知して学習を停止するための検証データの Loss と Accuracy を val_acc, val_loss である。学習を終了したモデルと評価用のテストデータで予測し、各モデルの正解率 (Accuracy) をそれぞれ Binary-Class 分類は表 7, Multi-Class 分類は表 8 にまとめる。また、実験は検証データを変えて 2 回行い、推移は全結果、予測結果は平均の値を表示する。

まず、Binary-Class 分類について述べる。図 7, 図 8 から、line_attn は初期から Accuracy, Loss 共に変化が少ないが、val_loss が僅かに減少しており、他より多い step 数で学習が停止している。しかし Accuracy は低いことから、学習できていないことが分かる。また bilstm_attn は、line_attn に比べて BiLSTM を用いていることで文脈情報が加味することができるが、train_loss が減少した際に、val_loss が上昇し過学習と検知して停止している。一方、BERT を用いている Model-A, Model-A_SUB, Model-B, Model-B_SUB はおよそ 40 step から val_loss が増加に転じ、学習が停止している。その際、train_loss は変わらず減少し、train_acc は増加していることから過学習が発生していることが分かる。また、表 7において、Model-A と Model-B は約 0.1 ポイント差で同程度の Accuracy だが、サブトレーニングタスクを用いた Model-A_SUB は Model-A から約 0.6 ポイントの減少、Model-B_SUB は Model-B から約 1.3 ポイントの増加となった。

次に、Multi-Class 分類について述べる。図 9, 図 10 から、line_attn は Binary-Class 分類と同様に学習できていないが、bilstm_attn は過学習を起こさず train_loss, val_loss 共に減少し、val_loss の減少が止まったことで学習が終了している。また、合わせて val_acc も向上していることから適切に学習できていることが分かる。

一方、BERT を用いている Model-A, Model-A_SUB, Model-B, Model-B_SUB は Binary-Class と同様に 40 step 程度で過学習が発生して学習が停止している。また、表 8 において、Model-A が Model-B よりも約 4 ポイント高い Accuracy を出しており、サブトレーニングタスクを用いた Model-A_SUB は Model-A から約 3.1 ポイントの増加、Model-B_SUB は Model-

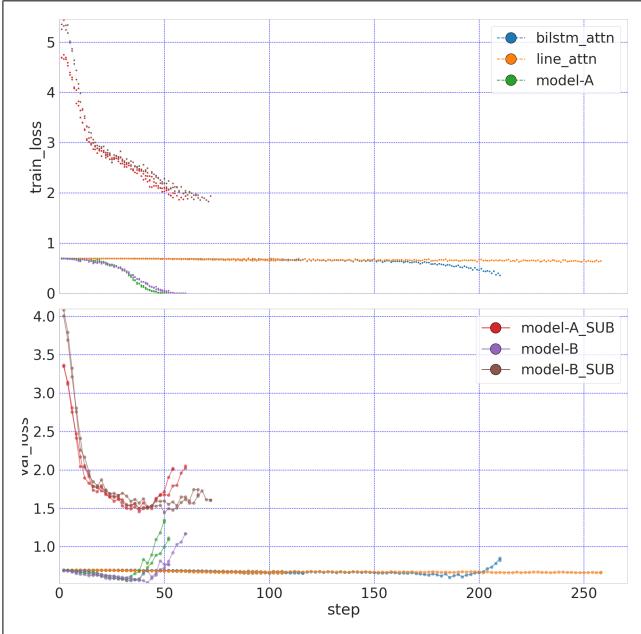


図 7: 含意関係認識 Binary-Class 分類: Loss の推移

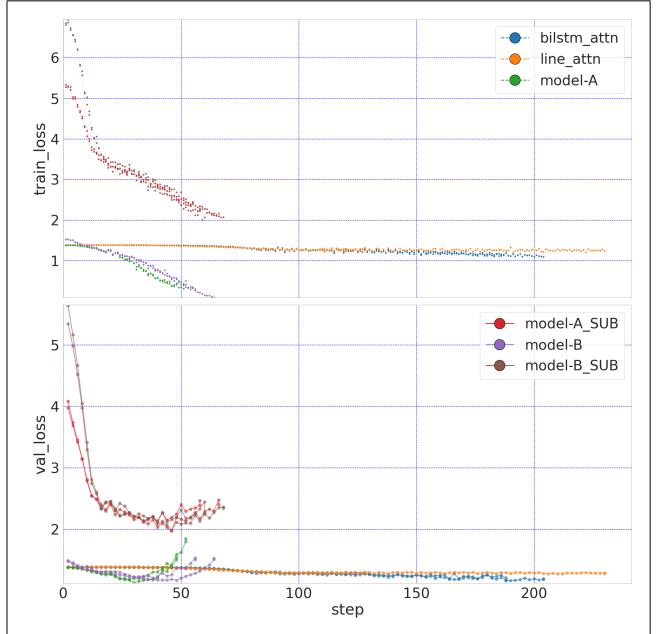


図 9: 含意関係認識 Multi-Class 分類: Loss の推移

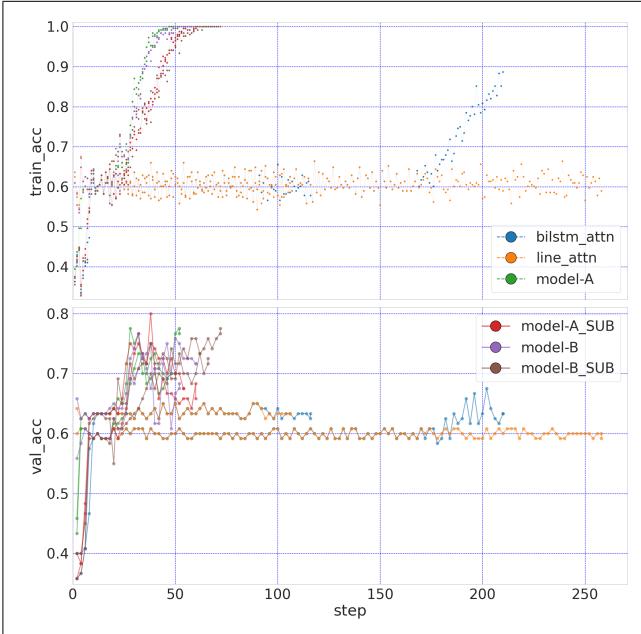


図 8: 含意関係認識 Binary-Class 分類: Accuracy の推移

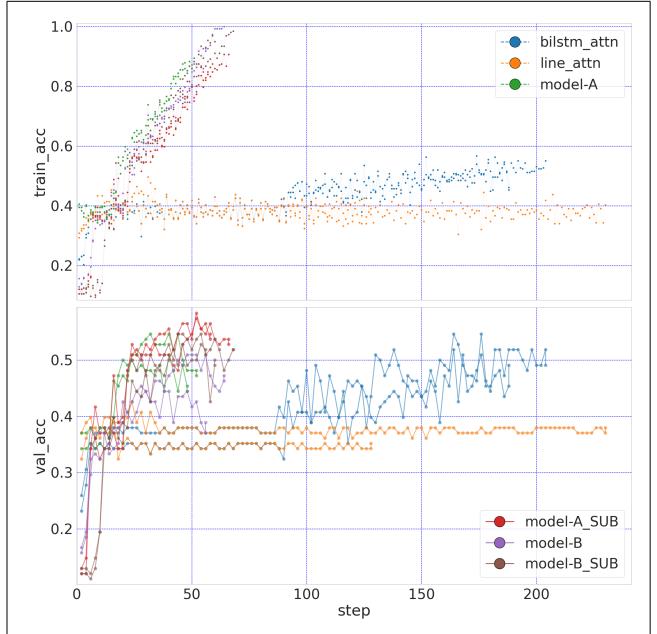


図 10: 含意関係認識 Multi-Class 分類: Accuracy の推移

B から約 4.2 ポイントの増加となった。さらに、Binary-Class 分類と異なり bilstm_attn の Accuracy も高く、Model-B を約 2.3 ポイント上回っている。

7.4 考 察

類似文予測の実験結果より、Model-A および Model-A-SUB, Model-A-SUB-attr の Pooling に Attention を用いた手法では、MLM, SOP, 属性損失 L_M を使うことで正解率 (Accuracy) が向上しており、複数の情報を学習に利用することで予測性能が向上することが確認できた。一方、Model-B および Model-B-SUB, Model-B-SUB-attr の Pooling において MEAN を行い Concatenation でマルチカテゴリ化した Model-B では、追

加した損失はいずれも有効でなく複数の情報を学習に利用するという点では、Model-A に劣る結果となった。

また、タスクを 1 つの属性としてモデルに適用し、複数属性を学習に用いないモデルの予測性能を考えると、類似文予測では Model-B、含意関係認識の Binary-Class 分類では Model-B-SUB, Multi-Class 分類では Model-A-SUB がそれぞれ高い Accuracy であった。このことから、多くの場合は MLM, SOP の損失を用いることで性能向上が見られ、Model-A と Model-B では、どちらが優れるかは決められず、さらなる検証が必要である。

表 7: 含意関係認識 Binary-Class 分類: テストデータ予測結果

Model	Accuracy
bilstm_attn	0.5803
line_attn	0.5803
model-A	0.6738
model-A_SUB	0.6672
model-B	0.6754
model-B_SUB	0.6885
baseline-NTCIR10	0.6393

表 8: 含意関係認識 Multi-Class 分類: テストデータ予測結果

Model	Accuracy
bilstm_attn	0.5091
line_attn	0.3741
model-A	0.5255
model-A_SUB	0.5560
model-B	0.4860
model-B_SUB	0.5286
baseline-NTCIR10	0.4544

8 おわりに

本稿では、ニューラルネットワークのモデルの出力する分散表現により多くの情報をマッピングし、入力文に付随するタスクのアノテーション以外の情報を学習に用いるマルチカテゴリ分散表現生成モデルおよびその学習法を提案した。具体的には、入力文に付随する情報を属性として定義し、BERT を用いてトークンベクトルを生成し、トークンベクトルから Attention を用いて属性カテゴリごとに分散表現を得る Model-A と、トークンベクトルの平均ベクトルを用い、属性カテゴリの埋め込みベクトルと統合することにより属性カテゴリごとに分散表現を得る Model-B を提案した。また、SOP(Sentence Order Prediction) と MLM(Masked Language Model) の損失と属性の比較から属性損失を定義し、それらを用いた学習法を合わせて提案した。その評価のため、類似文予測と含意関係認識において、Model-A と Model-B にそれぞれ SOP, MLM の損失を加えて学習したモデルとそうでないモデルを比較した。さらに、類似文予測では、属性損失を加えて学習したモデルも比較した。

実験の結果、Model-A については、類似文予測で複数属性を属性損失として用いることで性能が向上することを確認したが、Model-B では確認できなかった。また、Model-A の含意関係認識の Binary-Class 分類と Model-B の類似文予測以外のタスクで SOP, MLM の損失を用いることで正解率が向上した。以上のことから、MLM, SOP の損失を学習に用いることで多くの場合、性能の向上が見込めることが、また Model-A は Model-B より複数属性を用いることに向いていることが分かった。しかし、タスクごとの Model-A, Model-B の性能の差は、そのタスクやデータに依存するため、さらなる検証が必要である。

謝 辞

本稿では国立情報学研究所の IDR データセット提供サービスにより楽天株式会社から提供された「楽天データセット」を利用した。ここに記して感謝する。

文 献

- [1] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, 2014.
- [2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, 2019.
- [3] Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 670–680, 2017.
- [4] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 3982–3992, 2019.
- [5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, Vol. 30, pp. 5998–6008, 2017.
- [6] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*, 2020.
- [7] Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 1112–1122, 2018.
- [8] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 632–642, 2015.