既知語との表層類似性に基づく 未知語の分散表現の計算

修士論文

既知語との表層類似性に基づく
未知語の分散表現の計算
Computing embeddings for unknown words
using their surface similarity to known words

東京大学大学院 情報理工学系研究科
電子情報学専攻
48-186405　福田 展和

指導教員　　喜連川 優 教授

2020 年 1 月 30 日 提出

本論文は東京大学大学院情報理工学系研究科に修士号授与の要件として提出した修士論文である.

要約

現在、自然言語処理で標準的に用いられる深層学習モデルでは利用できる語彙が制限されるため、語彙に含まれない未知語 (Out of vocabulary: OOV) が問題となりやすい。この問題に対し、大規模テキストから言語モデルなどの補助的なタスクを介して、広範な語彙の単語埋め込みを事前に学習し、モデルの埋め込み層に固定して利用するアプローチが用いられる。しかし実応用では、モデルの訓練データより未来のテキストを処理したり、訓練データと異なるドメインのテキストを処理することも多く、新語や低頻度な合成語、表記ゆれ、綴り誤りなどの多様な未知語に対処することは困難である。

そこで本稿では、多様な未知語に対処するために、表層の類似する既知語を利用して未知語の埋め込み表現を計算する手法を提案し、表層の情報を利用する手法と文脈の情報を利用する手法に導入する。表層の情報を利用する手法では、既知語と未知語との間の表層の類似性を学習して、既知語の埋め込み表現を集約し、未知語の埋め込み表現を計算する。また、文脈の情報を利用する手法では、未知語が出現した文脈に加えて、表層の類似する単語の文脈を利用して、未知語の埋め込み表現を計算する。

実験では、表層を用いる提案手法と既存手法 [1, 2] により計算される未知語の埋め込み表現を、低頻度語の類似度判定データセット [3] と綴り誤りコーパス [4] を用いた内的評価で比較した。実験の結果、固有名詞や綴り誤りについてより良い埋め込み表現を計算できていることを確認した。また、未知語が多く含まれるツイッタードメインの品詞タグ付けと固有表現抽出による外的評価を通して、文脈を用いた既存手法 [5] とも比較を行った。実験の結果、提案手法によって未知語に対する分類において精度が改善することを確認した。

目次

図目次

表目次

第 1 章

序論

1.1 背景

マイクロブログや SNS などのコミュニケーション基盤が普及し、個人の情報発信が世界的に盛んになり、計算機が利用可能なテキストデータが急激に増加し続けている。その結果、社会的イベントに関する情勢を知りたい場合や、個人が発信した雑駁な情報を構造化したい場面において、構文解析や情報検索などの言語処理技術の重要性が増している。このような新たに生成され続けるテキストデータの処理における課題として、解析対象のテキストに新たな概念に応じて語彙や単語の表現が変化し続ける点が挙げられる。例えば、ソーシャルメディア上では、日々新しい概念に関する議論がなされており、それに伴い新しい単語や新しい表現が絶え間なく生まれ続けている。

また、英語を始めとする主要な言語では、様々なドメインのテキストをより高度に処理するために、新聞記事などの整った文書から構成されるコーパスだけでなく、科学技術論文や生物医学に関するテキストなどがデータとして整備されてきた。しかし、比較的話者の少ない言語では様々なドメインのテキストデータが未だ整備されていないため、英語と比較して十分なデータを利用できないという問題がある。このような異なるドメインのテキストデータを処理する場合の課題として、一般的なコーパスに存在しない専門的な概念や文法に伴った専門用語や表現などが多く含まれている点が挙げられる。例えば、生物医学に関するテキストでは、疾患・遺伝子・薬品などに関する専門用語が多く含まれており、同じ医療分野であっても薬学や疫学などの異なる分野間では出現する語彙が大きく異なる。このようなテキストデータを処理する場面では、システムによる解釈が困難なこれらの単語を適切に処理する必要がある。

言語処理システムを実世界に適用する上で、モデルの訓練データより未来のテキストを処理する状況や、異なるドメインのテキストを処理するような状況では、上述のように一般的なコーパスに出現しないような、新語や低頻度な合成語、表記ゆれ、綴り誤りなどの多様な未知語が多く出現する。しかしながら、自然言語処理で標準的に用いられる深層学習モデルでは利用できる語彙が制限されるため、語彙に含まれない未知語 (Out of vocabulary: OOV) が問題となりやすい。特にモデルを学習する訓練データが小規模である場合や、訓練データとテストデータのドメインが異なる場合では、未知語が大量に出現する。未知語は単語埋め込みの訓練時に出現しないため、未知語に対して言語処理モデルが適切な埋め込み表現を与えることが難しい。そのため、未知語の埋め込み表現の精度が低くなり、多くの未知語が出現するタスクでモデルの性能が低下する。この問題に対し、大規模テキストから言語モデルなどの補助的なタスクを介して広範な語彙の単語埋め込みを事前に学習し、モデルの埋め込み層に固定して利用するアプローチが用いられる。しかし、既存の単語埋め込み手法 [6, 7] では訓練対象の単語が訓練コーパス中に十分な回数観測されることを想定しており、単語埋め込みの訓練コーパスに出現しない未知語に対しては高精度な埋め込み表現を得ることができない。訓練コーパスのサイズを増やす、新しいコーパスを用いて訓練を行う等の方法では、多様な未知語を事前に網羅的に学習することは困難である。このような未知語を、事前に単語埋め込みが計算された既知語と同様に扱うためには、既知語の埋め込み表現の空間における未知語の埋め込み表現を計算する必要がある。

1.2　本研究の目的と貢献

本研究では、未知語の表層と文脈の情報を手がかりにした未知語の埋め込み表現の計算に取り組む。そのため、未知語と表層の類似する既知語を利用して未知語の埋め込み表現を計算する手法を提案する。まず、表層の情報を利用する手法では、対象となる未知語の表層に含まれる既知語や、表層の類似する既知語の埋め込み表現を集約することで未知語の埋め込み表現を計算する。既存手法では単語に含まれる文字や部分文字列であるサブワードに割り当てられた埋め込み表現から元の単語埋め込みを模倣する。一方、提案手法では表層の類似する既知語の間の関係性を表層の情報に基づいて学習する。次に、文脈の情報を利用する手法では、未知語が出現する文脈に加えて、

表層の類似する単語の文脈を利用して、未知語の埋め込み表現を計算する。

本稿では、内的評価と外的評価を通して提案手法の未知語の埋め込み表現の評価を行った。内的評価として、低頻度語の類似度判定タスク [3] と綴り誤りコーパス [4] から抽出した綴り誤りの埋め込み評価タスクを用いて、表層の情報を利用する手法を評価した。外的評価として、未知語が多く含まれるドメインの品詞タグ付けと固有表現抽出を用いて、表層を利用する手法と文脈を利用する手法を評価した。実験の結果、提案手法が計算する未知語の埋め込み表現の性能が既存手法と同程度もしくは上回ることを確認した。

1.3　本論文の構成

本論文の構成は以下の通りである。

第 2 章 単語埋め込みと類似文字列検索に関して、本論文の前提となる基本的な事項を説明する。

第 3 章 未知語の埋め込み表現に関して、本論文と関連する手法について説明する。

第 4 章 既知語との表層の類似性を手掛かりとして未知語の埋め込み表現を計算する提案手法について説明する。

第 5 章 表層を利用する提案手法と既存手法を比較するために行った、未知語の埋め込み表現の内的評価について説明する。

第 6 章 表層と文脈を利用する提案手法と既存手法を比較するために、各手法によって計算された未知語の埋め込み表現を用いた、下流タスクの外的評価について説明する。

第 7 章 本稿のまとめと今後の課題について説明する。

第 2 章

事前知識

2.1 単語分散表現手法

現在、自然言語処理において標準的に用いられる深層学習モデルは、言語における離散的なシンボルである単語にその意味を表す数十から数百次元の密で連続的な実数値ベクトル表現を割り当てて扱う単語分散表現手法に基づいている。この単語の意味表現としてのベクトル表現は分散表現や埋め込み表現と呼ばれ、「ある語の意味はその周辺に存在する語の分布によって特徴付けられる」という分布仮説 [8] の考えに基づいている。その中でも、Mikolov ら [6] の手法を始めとしたニューラルネットワークに基づく手法によって、大規模なテキストコーパスから学習された単語分散表現が様々な自然言語処理タスクにおける性能向上に貢献してきた。この単語分散表現手法において代表的な手法である、Skip-gram、GloVe、fastText を順に説明する。

Skip-gram まず、Skip-gram with negative sampling (SGNS) モデル [6] について説明する。Skip-gram では訓練データのテキストコーパスに含まれる学習対象となる語彙の各単語に対して 1 つの分散表現を割り当てる。次に、以下に説明する損失関数に従ってこれらの分散表現を学習する。まず、大規模なテキストコーパスの文章中に表れる単語系列を $w_1, w_2, \cdots, w_n$ とする。この単語列の中で、ある単語 w_t に注目したとき、対象単語 w_t を中心として一定のウィンドウ幅 wnd 内の周囲の単語 $w_c \in C_t = \{w_{t-wnd}, \cdots, w_{t-1}, w_{t+1}, \cdots, w_{t+wnd}\}$ を文脈単語とする。Skip-gram モデルは中心単語から周囲の文脈単語を推定し、以下のように共起確率を最大化するように学習される。

$$P(w_{t-wnd}, \cdots, w_{t-1}, w_{t+1}, \cdots, w_{t+wnd}|w_t) \tag{1}$$

すなわち、中心の対象単語から周囲の文脈単語を良く推定できるように各単語の単語分散表現が学習される。この共起確率を計算するために、対象単語と文脈単語の組からそれらの共起の度合いを計算する関数 $s(w_t, w_c)$ を導入する。ここで $\boldsymbol{u}_t, \boldsymbol{v}_c$ はそれぞれ単語 w_t, w_c に割り当てられた分散表現である。

$$s(w_t, w_c) = \boldsymbol{u}_t^{\mathsf{T}} \boldsymbol{v}_c \tag{2}$$

上記の文脈単語の共起確率は $s(w_t, w_c)$ を用いて以下のように定義される。ここで W は語彙を表す。

$$P(w_c|w_t) = \frac{\exp(s(w_t, w_c))}{\sum_{c' \in W} \exp(s(w_t, w_{c'}))} \tag{3}$$

しかし、この計算は語彙 W に含まれる全ての単語に対して softmax 関数を計算する必要があるため効率的でない。そこで、negative sampling は語彙 W の部分集合 N_t に対して softmax を計算し近似することで効率的に計算を行う。ここで $l(x) = \log(1 + \exp(-x))$、$N_t$ は語彙からランダムにサンプルした単語集合である。

$$l\left\{s(w_t, w_c)\right\} + \sum_{n \in N_t} l\left\{-s(w_t, w_n)\right\} \tag{4}$$

この近似計算は以下のように解釈できる。上述の中心単語から各文脈単語を独立に推定するタスクは文脈単語が存在するかしないかを推定する識別問題と見なすことができる。この識別問題において、テキストコーパスにおいて中心単語の周囲に出現する文脈単語は正例であり、語彙からランダムに選択した単語を負例の文脈単語として扱う。このように文脈単語の共起確率が計算され、最終的な損失関数は以下のようになる。

$$J = \sum_t \left[\sum_{c \in C_t} l\left\{s(w_t, w_c)\right\} + \sum_{n \in N_{t,c}} l\left\{-s(w_t, w_n)\right\} \right] \tag{5}$$

GloVe SGNS の他に代表的な単語分散表現手法として GloVe [7] が提案されている。Skip-gram 等の局所的なウィンドウ内での文脈を学習するモデルは、コーパス中の大域的な情報を上手く捉えられないことが指摘されている。この問題に対処するために、Skip-gram のような周囲の単語を推定するタスクを学習するのではなく、GloVe は単語の共起行列から単語分散表現を学習することで大域的な情報を獲得する。まず、コー

パス中に出現する、ある単語とその文脈に出現する単語対の共起行列を計算する。次に、Skip-gram の損失関数を以下のように変更して単語分散表現を学習する。ここで $X_{i,j}$ は共起行列の非ゼロ要素、$f(x)$ は重み関数である。

$$J = \sum_{i,j}^{W} f(X_{i,j}) \left(\boldsymbol{u}_j^{\mathsf{T}} \boldsymbol{v}_i - \log X_{i,j} \right)^2 \tag{6}$$

この損失関数によって局所的な情報であるウィンドウ内の共起スコアと大域的な共起行列の二乗誤差を最小化する。

fastText 上記のモデルは形態素の情報を無視しており、単語に含まれる形態素に関わらず全単語を離散的なシンボルとして分散表現を訓練する。この問題に対処するため、fastText [9] は表層の情報を利用しながら単語分散表現を学習する。基本的に fastText は Skip-gram モデルを拡張した手法であり、単語に含まれるサブワードの情報を利用する点が異なる。まず、各単語は文字 n-gram の集合に加えて単語自身から表現される。このとき、接頭辞や接尾辞を区別するために境界となる記号を加える。例えば単語"where"を例として $n = 3$ とした場合には、文字 n-gram は<wh, whe, her, ere, re>となり、これに<where>を追加した集合から構成される。そして、Skip-gram では各単語に分散表現が割り当てられるが、fastText では各サブワードに分散表現が割り当てられる。そして、単語に含まれる n-gram の分散表現の和によって単語を表現する。すなわち、中心単語と文脈単語の共起度を表す関数 $s(w_t, w_c)$ を以下のように修正する。

$$s(w_t, w_c) = \sum_{g \in G_{w_t}} \boldsymbol{z}_g^{\mathsf{T}} \boldsymbol{v}_{w_c} \tag{7}$$

このように、サブワードの分散表現を用いて構成的に単語の分散表現を表すことによって、形態素の情報を用いてより高精度な単語分散表現を獲得することができる。また、サブワードの埋め込み表現の集約を行う Self-Attention [10, 11] を fastText に導入した手法 [12] も提案されている。

次に、サブワード単位の分かち書きを利用する手法について説明する。機械翻訳において、目的言語のサブワードを生成するモデルはその出力コストが語彙サイズに依存する。そのため、この計算コストを削減するために、事前に語彙を削減してからモデルの学習が行われる。このようにサイズの小さいサブワードの語彙を得る手法として、

代表的なものに Byte pair encoding (BPE) [13] と Unigram language model [14] がある。BPE [13] は語彙を 1 文字から開始し、各サブワードを連結した際に最も頻度が高くなる組を選び新たに語彙に追加する。このように、サブワードの結合を繰り返し語彙に追加する操作を決められた語彙サイズに達するまで続ける。得られた語彙による分割は、結合ルールと同じ順序で決定的に行われる。Unigram language model [14] はユニグラム言語モデルと EM アルゴリズムを用いてサブワードの語彙を削減する。まず、単語・サブワード・文字から構成される十分大きなサイズの語彙を用意する。次に各サブワードの出現確率から尤度が最大となる分割系列をビタビアルゴリズムを用いて計算する。そして得られた分割系列から、各サブワードの出現確率を更新する。さらに、各サブワードを語彙から削除した際の全体の尤度の変化を計算し、最も減少量の小さい一定数のサブワードを語彙から削除する。この操作を決められた語彙サイズを下回るまで繰り返すことによって語彙サイズを削減する。得られた語彙による分割は、ユニグラム言語モデルから確率的に得られる。これらの手法は主にサブワードの語彙を構築する手法であり、各サブワードの分散表現やそれらから構成される単語の分散表現を求める手法ではない。

最後に、文脈を考慮した単語分散表現手法について説明する。これらの手法は、テキストコーパスにおいて言語モデルなどの補助タスクを解くことによって、文脈を考慮した単語やサブワードの分散表現を学習する。Peters ら [15] は Long short-term memory (LSTM) [16] を用いた双方向の言語モデルを学習することによって文脈を考慮した単語分散表現 (Embeddings from Language Models: ELMo) を学習して下流タスクに転用することで様々な言語処理タスクの性能が向上することを示した。Devlin ら [17] は Transformer [10] のエンコーダを用いてマスクされた言語モデルを学習するモデル (Bidirectional Encoder Representations from Transformers: BERT) を提案した。これらの文脈を考慮した単語やサブワード単位の分散表現は、一般的に文を入力して LSTM や Transformer などのネットワークを通して分散表現を動的に計算する。文毎に動的に分散表現を計算するため、文脈を考慮した分散表現を計算することができる。特に多義語については、同じ単語であっても異なる周囲の文脈を利用することで、文脈に応じた異なる分散表現を出力することができる。

2.2　類似文字列検索

事前に大規模なテキストコーパスで学習した単語分散表現を、大規模なウェブテキストなどの実データに適用する場合には、表記ゆれや綴り誤りなどに起因して表層表現が一致しないという問題が存在する。そのため、本研究ではこの問題を解決するために表層の類似する文字列を高速にマッチングさせる手法を用いる。この問題に対処するために、ステミングや綴り誤り訂正などの手法が古くから研究されてきた。これらの手法では文字列を標準的な形に書き換える規則を人手や機械学習を用いて獲得しており、適用する言語やコーパスに強く依存する。

言語に依存しないより簡素な手法として類似文字列検索が挙げられる。類似文字列検索に関連して古くから用いれてきた手法に Levenshtein オートマトンや、ビットパラレル法などがある。以下では、「文字列の集合からクエリ文字列と類似する文字列を検索する操作」というタスクに限定して類似文字列検索を説明する。特に本稿では、文字列集合は単語分散表現の語彙（既知語）に対応し、クエリ文字列は単語（未知語）に対応する。

2.2.1　CPMerge [18]

[18] では、文字列から特徴量を抽出して、特徴量集合の集合類似度を用いて類似文字列検索を行う。また、類似文字列検索を τ オーバーラップ問題として定式化し、この問題に対して CPMerge アルゴリズムを提案した。

本研究では文字列の表層の特徴量として文字 tri-gram を用いる。例えば、文字列 $x = \texttt{spaghetti}$ という単語は以下のように文字 tri-gram からなる特徴量集合 X で表される。ここで、境界記号として文字列の先頭と末尾に \$ が挿入される。

$$X = \{\$\$\mathrm{s}, \$\mathrm{sp}, \mathrm{spa}, \cdots, \mathrm{tti}, \mathrm{ti}\$, \mathrm{i}\$\$\} \tag{8}$$

文字列間の類似度としてこのような特徴量集合に対する集合類似度を用いる。例えば、文字列 x と y に対応する集合 X と Y の間の Jaccard 係数は以下のように計算される。

$$\mathrm{jaccard}(X, Y) = \frac{|X \cap Y|}{|X \cup Y|} \tag{9}$$

この類似度が α 以上となる条件 $\mathrm{jaccard}(X, Y) \geq \alpha$ から以下の関係式が得られる。

$$|X \cap Y| \geq \frac{\alpha(|X| + |Y|)}{1 + \alpha} \tag{10}$$

$$\alpha|X| \leq |Y| \leq \frac{|X|}{\alpha} \tag{11}$$

ここで $|X \cap Y|$ は X と Y の最小オーバーラップ数 τ と定義される。式 10 から τ は $|X|, |Y|, \alpha$ に依存する値である。さらに、式 11 から検索対象となる文字列の $|Y|$ の範囲は、$|X|, \alpha$ から限定される。

上記の議論から類似文字列検索は、クエリ X と特徴を τ 個以上共有する Y を見つける τ オーバーラップ問題に帰着される。この問題を解くために、特徴をキーとしてその特徴を含む文字列を値に持つ転置インデックスを利用する。この転置インデックス上で、X に含まれる特徴のリストに τ 回以上出現する単語を見つけることで τ オーバーラップ問題を解くことができる。

CPMerge アルゴリズムではこの転置インデッククスを高速に検索するために主に解候補の生成と枝刈りに関する工夫をしている。Algorithm 1 に CPMerge の概要を示す。まず、式 11 に基づいて $|Y|$ の範囲を限定する。次に各 $|Y|$ について、X に含まれる特徴をキーとするリストを抽出して、リストのサイズの昇順に並べ替える。このとき、特徴集合 X から任意の $|X| - \tau + 1$ 個の特徴を選ぶと必ず解が含まれている、という性質を用いてリストのサイズが小さい方から順に、リスト内の各文字列が X と共有する特徴数 $M[i]$ を順番に数え上げる。この操作を $k \in \{0, \cdots, |X| - \tau\}$ の特徴まで順番に行い、解候補を得る。次に各解候補毎に X の残りの特徴 $k \in \{|X| - \tau + 1, \ldots, |X| - 1\}$ を持つか順番に走査してゆき $M[i]$ を数え上げる。このとき、τ の上限値が $\tau \leq M[i] + |X| - k - 1$ であることを利用して、$M[i]$ が $|X| + \tau + k + 1$ を下回ったものを枝刈りする。このように走査を行い、$M[i] \geq \tau$ を満たした文字列を最終的な解として加えてゆく。CPMerge はこのように類似文字列検索を高速に行う。

Algorithm 1 CPMerge

Input: D: 転置インデックス
Input: x: 検索クエリ文字列
Input: α
Output: R: 類似度が α 以上の文字列集合

1: x から特徴集合 X を作成する
2: 式 11 から $|Y|$ の範囲 $n, \cdots, N$ を計算する
3: $R \leftarrow []$
4: **for all** $l = n$ to N **do**
5:　式 10 から τ を計算する
6:　D から $|Y| = l$ となる転置リスト d を抽出する
7:　X の要素を d の要素数の昇順にソートする
8:　$M \leftarrow \{\}$
9:　**for** $k = 0$ to $|X| - \tau$ **do**
10:　　**for all** $i \in d[X[k]]$ **do**
11:　　$M[i] \leftarrow M[i] + 1$
12:　　**end for**
13:　**end for**
14:　**for** $k = (|X| - \tau + 1)$ to $(|X| - 1)$ **do**
15:　　**for all** $i \in M$ **do**
16:　　**if** $i \in d[X[k]]$ **then**
17:　　$M[i] \leftarrow M[i] + 1$
18:　　**end if**
19:　　**if** $M[i] \geq \tau$ **then**
20:　　i を R に追加し、M から削除
21:　　**else if** $M[i] + (|X| - k - 1) < \tau$ **then**
22:　　i を M から削除
23:　　**end if**
24:　　**end for**
25:　**end for**
26: **end for**
27: **return** R

2.2.2 Adaptive q-gram [19]

[19] では編集距離（レーベンシュタイン距離）の近い文字列を検索する上で q-gram 類似度を利用して、Top-k の類似文字列検索を行う高速な Branch and Bound アルゴリズムと Adaptive q-gram アルゴリズムを提案した。q-gram とは文字列に含まれる長さ q の部分文字列とその部分文字列の開始位置を組み合わせたものである。例えば、$q = 3$ のときの `university` の q-gram は、(1,`uni`), (2,`niv`), (3,`ive`), ⋯, (8, `ity`) である。q-gram 類似度とは二つの文字列が共有する q-gram の数である。編集距離と文字列に含まれる q-gram 類似度との間には以下の関係が成り立つことが知られている。P と S の編集距離が τ のとき、q-gram 類似度 t の下限値は以下の式で表される。

$$t = \max(|P|, |S|) - q + 1 - q\tau \tag{12}$$

また、文字列の長さと編集距離 $(ed(P, S))$ の間に以下の関係が成り立つ。

$$\mathrm{ed}(P, S) > |P| - |S| + 1 \tag{13}$$

Top-k の類似文字列検索も τ オーバーラップ問題と同様に、q-gram をキーとして q-gram を含む文字列の集合を値に持つ転置インデックス上を検索することによって、問題を解いてゆく。

Branch and Bound (BB) アルゴリズムでは、式 12 から文字列の長さと q-gram 類似度の閾値から検索を効率的に枝刈りを行う。ここで、検索対象となる文字列集合に含まれている文字列を S、クエリ文字列を P とする。BB アルゴリズムでは、まず q-gram 類似度の閾値を 1 に初期化する。次に、P と S の長さの差の昇順に転置リストを走査し、解を見つける度に q-gram 類似度の閾値を式 12 に基づいて更新してゆく。そして式 13 から、S の長さに基づいて検索を打ち切る。

Adaptive q-gram (AQ) アルゴリズムでは上記のアルゴリズムに加えて、複数の q の値を用いることでより効率的な検索を行う。Algorithm 2 に AQ アルゴリズムの概要を示す。BB アルゴリズムでは単一の q のみを用いていたが、q の値が小さい場合にはリストに含まれる文字列の数が多くなるため頻度の数え上げのコストが大きくなる。そこで、転置リスト内に要素が少なくとも 1 つは含まれており、かつ出来るだけ大きな q を選択することで、リスト内の要素数を減らすことができる。このようなアルゴリズムを用いることで、より効率的な Top-k の類似文字列検索を行うことができる。

Algorithm 2 Adaptive q-gram Algorithm

Input: T: 転置インデックス

Input: P: クエリ文字列

Input: k

Output: Q: Top-k 文字列

1: $q'_{\text{top}-k} = q'_{\min}$
2: $t'_{\text{top}-k} = 1, ld = 0$
3: **while** TRUE **do**
4: 　T から $q'_{\text{top}-k}$ に関する転置リスト T' を抽出
5: 　P から $q'_{\text{top}-k}$-gram を抽出
6: 　T' から $||S| - |P|| = ld$ である転置リスト I を抽出
7: 　$t'_{\text{top}-k}$ に基づいて I を走査して q-gram 類似度を数える
8: 　解候補をソートして Top-k 文字列 Q を計算
9: 　$\tau'_{\text{top}-k} = \text{ed}(P, Q)$
10: 　**if** $\text{ed}(P, Q) \leq (ld + 1)$ **then**
11: 　　break
12: 　**end if**
13: 　$ld = ld + 1$
14: 　$q'_{\text{top}-k}$ を更新
15: 　式 12 から $t'_{\text{top}-k}$ を更新
16: **end while**
17: **return** Q

これらの類似文字列検索のアルゴリズムは人手で設計されたルールや、訓練データを用いないため、どのような言語やドメインの語彙にも適用することが可能である。

第 3 章

関連研究

本章では本手法に関連する研究について説明する。Skip-gram を始めとする 1 つの単語に 1 つの分散表現を与える手法は、訓練コーパスに出現しない低頻度語や未知語に対して精度の良い埋め込み表現を得ることができない。これらの手法は訓練対象の単語が訓練コーパス中に十分な回数観測されることを想定しており、訓練コーパスに出現しない未知語については埋め込み表現を得ることができないためである。また、訓練コーパス中にほとんど出現しない低頻度語についても、その単語の文脈を十分に訓練することができないため分散表現の性能が低いと考えられる。大規模なテキストコーパスで訓練した単語分散表現を下流のタスクに利用する場合に、これらの低頻度語や未知語が下流タスクのテストデータに出現する際には、低頻度語や未知語に対応する分散表現を利用できないためにタスクの性能が低下すると考えられる。このような低頻度語や未知語は単語分散表現を事前訓練する際に、十分大きなテキストコーパスを用いて訓練し、より多くの単語を網羅することによって対処することができると考えられる。しかし、訓練コーパスのサイズやコーパスの語彙サイズは有限である。造語や綴り誤りや表記ゆれ、低頻度な合成語などの全ての低頻度語や未知語の分散表現を事前に学習することは困難である。そのため、訓練コーパスの語彙に含まれない未知語に対して、既知語と同じ意味空間の埋め込み表現を生成する必要がある。

文脈を考慮して単語やサブワードの分散表現を生成するモデルにおいても、文字列を分散表現に対応付けるために埋め込み表現が利用されている。例えば、ELMo では文字単位の埋め込み表現が、BERT ではサブワード単位の埋め込み表現が利用されている。これらの手法は、fastText と同様にサブワード等の埋め込み表現を利用することで未知語に対しても分散表現を与えることができるが、既知語と未知語の明確な区

別が存在しないため、低頻度語や未知語に対する性能を比較することが難しい。しかし、これらのモデルにおいても、事前に言語モデルを訓練するテキストコーパスに出現しない低頻度語や未知語については、その分散表現の性能が低いという同様の問題が存在すると考えられる。また、Pruthi ら [20] は BERT が利用する BPE 等のサブワード単位の分かち書きモデルが敵対的な表層の変化に影響を受けやすいことを報告している。本手法の対象は単語単位の表記ゆれや綴り誤りであるが、本手法をサブワード単位の分かち書きを利用する手法と組み合わせることでこのような敵対的な表層変化にも対処できると考えられる。

上述の低頻度語や未知語の問題に対処するために、様々な手法が検討されてきた。低頻度語や未知語の分散表現を推定する手法として主に、表層の情報を利用する手法、文脈の情報を利用する手法、外部知識を利用する手法が提案されている。以下に各手法について述べる。

3.1　表層を用いる手法

低頻度語や未知語の単語分散表現を推定する際に表層の情報を利用する手法が広く研究されてきた [12,21–25]。これらの手法の中でも主に、サブワードの埋め込み表現を利用する手法 [1,2,12] が広く用いられている。これらの手法は単語を文字やサブワードに分解してから分散表現を再構成することで、事前に訓練された単語分散表現を模倣する。Pinter ら [26] は文字分散表現を LSTM に通して単語分散表現を再構成する手法を提案した。Zhao ら [1] は Bojanowski [9] らの手法と同様にサブワード分散表現の和によって単語分散表現を再構成する手法を提案した。具体的には、単語に含まれる長さ 3-6 の文字 n-gram のサブワードを列挙し、以下の式のように各サブワードに割り当てたサブワード埋め込みの平均を計算することで単語の埋め込み表現を計算する。ここで、$\mathcal{S}(w)$ は単語 w に含まれるサブワードの集合であり、$\boldsymbol{v}_s$ はサブワード埋め込みである。

$$\hat{\boldsymbol{e}}_w = \frac{1}{|\mathcal{S}(w)|} \sum_{s \in \mathcal{S}(w)} \boldsymbol{v}_s \tag{14}$$

このように計算した単語埋め込みと元の埋め込み表現との損失関数を用いてサブワード埋め込みを更新する。

$$\mathcal{L} = ||\hat{\boldsymbol{e}}_w - \boldsymbol{e}_w||^2 \tag{15}$$

この手法は語彙中の全ての単語に含まれるサブワードを列挙して、各サブワードに 1 つの埋め込み表現を与えるため、埋め込み表現に必要な容量が大きいという問題がある。例えばオリジナルの fastText[*1]に含まれる約 200 万の語彙から長さ 3-6 の文字 n-gram を全て列挙するとそのサブワード数は 620 万になる [2]。分散表現の容量で比較すると、元の埋め込み表現が約 2.2GB であるのに対して、サブワード埋め込みは 7.1GB にも及ぶ。この問題を解決するために、Sasaki ら [2] は複数のサブワード間で埋め込み表現を共有する手法を提案した。具体的には、サブワードから埋め込み表現へのマッピング関数として、以下の式のように頻度上位 F 件のサブワードのみを用いる関数と、ハッシュ関数を用いてサイズ H の埋め込み表現を共有する関数を導入した。ここで、η はサブワードから埋め込み表現のインデックスへのマッピング関数、S はサブワードの語彙、$\mathcal{S}_F \subseteq \mathcal{S}$ は頻度上位 F 件のサブワードの集合を表す。

$$\eta_F(\cdot) : \mathcal{S}_F \to \mathcal{I}_{s,F} \text{ where } \mathcal{I}_{s,F} = \{1, \ldots, |\mathcal{S}_F|\} \tag{16}$$

$$\eta_H(\cdot) : \mathcal{S} \to \mathcal{I}_{s,H} \text{ where } \mathcal{I}_{s,H} = \{1, \ldots, H\} \tag{17}$$

これらのマッピング関数は、埋め込み表現のベクトルを $\boldsymbol{V}$ として $\boldsymbol{v}_s = \boldsymbol{V}[\eta(s)]$ のように用いられる。このように、サブワード数と埋め込み表現の数を限定することで、容量を削減しながらもより高精度な未知語の埋め込み表現を計算することを目的としている。さらに、サブワード埋め込みの集約方法として、以下の式のように Self attention [10] を利用して単語埋め込みを再構成する手法を提案した。ここで、Z はス

[*1] https://fasttext.cc/docs/en/english-vectors.html

ケーリングパラメータ、$\boldsymbol{q}_s, \boldsymbol{k}_s, \boldsymbol{v}_s$ はサブワード埋め込みである。

$$\hat{\boldsymbol{e}}_w = \sum_{s \in \mathcal{S}(w)} a_{s,w} \boldsymbol{v}_s \tag{18}$$

$$a_{s,w} = \frac{\exp(Z\hat{\boldsymbol{q}} \cdot \boldsymbol{k}_s)}{\sum_{s' \in \mathcal{S}(w)} \exp(Z\hat{\boldsymbol{q}} \cdot \boldsymbol{k}_{s'})} \tag{19}$$

$$\hat{\boldsymbol{q}} = \sum_{s \in \mathcal{S}(w)} \boldsymbol{q}_s \tag{20}$$

これらの手法は、学習済みの既知語の単語分散表現を訓練データとして、単語をサブワード等の要素に分解し、それらの分散表現から元の単語分散表現を推定する再構成タスクを学習している。サブワード等に分解することによって、元の単語分散表現では分散表現を生成できなかった未知語の分散表現を推定することができる上に、元の単語分散表現に比べて保存する分散表現の数を削減することができる。利用するベクトルの数と単語埋め込みの性能は比例することが報告されている [2]。しかし、これらの手法は単語の埋め込み表現を学習した後に、サブワード等の埋め込み表現を学習する必要がある。また、利用するベクトルの数に比例して学習コストが増大し、2 段階のパイプライン学習のためにモデルが生成する分散表現は、学習に用いられた既知語の分散表現を復元する上で誤差を生じるという問題がある。

また、綴り誤りに対して頑健な埋め込み表現に関する研究として、Edizel ら [27] によって fastText を拡張して、綴り誤りに頑健な単語埋め込みの学習手法が提案されている。この手法では、ウェブ上の検索クエリのログを用いて現実的な綴り誤りのデータセットを構築し、これを用いて fastText を訓練する。しかし、この手法は綴り誤りのデータセットを用意する必要があり、大規模な検索ログを収集できない言語に適用することが困難である。また、Mizumoto ら [28] は全ての綴り誤りが必ずしも下流タスクの精度を低下させず、スペル訂正を行わなくとも接辞の情報を用いることで十分精度を維持することができることを示した。本稿では埋め込み表現の外的評価として、接辞などの表層の情報を利用する各手法の性能を下流タスクを通して評価する。

3.2　文脈を用いる手法

低頻度語や未知語が出現した周囲の文脈の情報から埋め込み表現を推定する手法が研究されてきた [29–34]。これらの手法は、低頻度な単語が出現する限られた文脈の情報から高品質な埋め込み表現を学習することを目的としている。Herbelot ら [29] は、Skip-gram モデルを拡張し低頻度語の埋め込み表現を更新する手法を提案した。Lazaridou ら [35] は、周囲の単語埋め込みを平均することで対象の単語の埋め込み表現を計算する手法を提案した。Kohdak ら [36] は、Lararidou らの方法で計算した単語埋め込みの線形変換が性能を向上することを示した。これらの低頻度語の単語分散表現の推定は、Few-shot の回帰問題であると見なすことができる。そのため、推論時に低頻度語が出現する複数の文脈からその分散表現を推定することを考え、ある既知語についてその単語が出現する文脈を入力としてその単語の分散表現を推定するように学習する。これらの手法は単語の分散表現を学習できる程度の文脈を利用可能であることを想定しており、周囲の文脈がほとんど利用できない場合には適用できない。

これらの文脈の情報を利用する手法は表層の情報を利用する手法と組み合わせることができる。Schick ら [37] は Kohdak ら [36] の線形変換を用いるモデルに文脈の情報を読み取る attention を導入し、サブワードの分散表現から計算した表層の情報を組み合わせるモデルを提案した。Hu ら [5] は Self-attention のエンコーダブロック [10] を用いて低頻度語の文脈をエンコードするし、文字 Convolutional neural network (CNN) を通して計算した表層の情報と組み合わせる手法を提案した。Peng ら [38] は Schick ら [37] の手法を固有表現抽出に適用し、低頻度語の分散表現を計算するモデルを Student、固有表現抽出モデルを Teacher と見なして、Teacher-student の枠組みでモデルを学習する手法を提案した。

上述した低頻度語や未知語の分散表現を推定するモデルにおいて、表層の情報を利用する場合には文字やサブワードの分散表現を基に単語埋め込みを再構成する手法が用いられている。本稿では、サブワード分散表現を用いて埋め込み表現を再構成する手法をベースラインとして比較する。また、文脈の情報と提案手法による効果が相補的であることを実験によって確認する。

3.3　外部知識を用いる手法

低頻度語や未知語の単語分散表現を推定する際に外部知識を利用する手法が研究されてきた [39–42]。これらの手法は主に外部リソースを利用し、未知語の定義文や対応する知識グラフ上のエンティティなどを参照する。Bahdanau ら [43] は LSTM を利用して WordNet の単語定義文から単語分散表現を推定する手法を提案した。Pilehvar ら [44] は WordNet 上で意味の類似する単語を利用して未知語の分散表現を推定する手法を提案した。Yang ら [45] は知識グラフを利用し、Graph Convolutional Network を通して未知語の単語分散表現を推定する手法を提案した。これらの手法は、未知語の中でも特に表層から意味を類推することが困難な新語などに対して有効であると考えられる。しかし、これらの手法によって分散表現を推定する際には、対象となる未知語を外部リソースと対応付ける必要があり、それに応じた問題が存在する。まず、対象となる未知語の表層が外部知識のエンティティの表層と一致する必要がある。例えば、表記ゆれや綴り誤りが存在する場合には外部知識への対応付けが自明でない問題となる。さらに、同じ表層を持つ異義語が存在する場合には、未知語が出現する文脈と各エンティティの意味を対応付けて判別する必要などがあり、単語分散表現を生成する他の手法と比較するとコストが大きい。また、未知語が外部知識に登録されていない場合にも適用することができない。

第 4 章

提案手法

本章では未知語埋め込み表現を計算する提案手法について述べる。以下に、表層の類似する既知語の埋め込み表現を利用する手法、表層の類似する単語の文脈を利用する手法について順に説明する。

4.1 表層の類似する既知語の埋め込み表現を利用する手法

本節では表層の情報から未知語の埋め込み表現を計算する手法について説明する。まず、**BoS** [1] を例として、サブワード埋め込みから未知語の単語埋め込みを計算する場合の問題点について述べる。**BoS** で用いられるサブワードの埋め込み表現は、このサブワードを含む全ての単語の意味を復元するように学習されるため、単語の埋め込み表現と比較してサブワードの埋め込み表現は曖昧になる場合があると考えられる。例として、`higher` の埋め込み表現を計算する場合には、表層に含まれる `high` や `er` などのサブワードを手掛かりとすることが意図されている。しかし、`higher` に含まれるサブワード `er` は `never, member` などの単語にも含まれており、多義性が高いため単独のベクトルでその意味を捉えることは難しい。例として、単語 `higher` に対して **BoS** [1] を用いて学習したサブワードの埋め込み表現と元の単語の埋め込み表現との関係について説明する。まず、単語 `higher` は境界記号`<>`が付与され、以下のような長さ 3-6 の文字 n-gram のサブワードに分解される。

$$\texttt{<hi}, \texttt{hig}, \cdots, \texttt{er>}, \texttt{<hig}, \texttt{high}, \cdots, \texttt{her>}, \texttt{<high}, \texttt{highe}, \cdots, \texttt{gher>}$$

次に、これらのサブワードに埋め込み表現を割り当て、元の単語の埋め込み表現を模倣するように、サブワード埋め込みを学習する。このように学習したサブワードの埋め

サブワード	highe <high high er>	**BoS**
コサイン類似度	48.4　34.2　20.1　−7.8	36.5

既知語	high
コサイン類似度	69.8

表 1　higher のサブワード埋め込みの例。

込み表現と元の単語の埋め込み表現との類似度をコサイン類似度で計算した。表 1 に higher を構成するサブワードの一部の埋め込み表現と元の単語の埋め込み表現との関係を示す。なお、単語埋め込みには **GloVe** [7] を用いた。表から、サブワードの中で最も元の埋め込み表現と近いサブワード highe であってもそのコサイン類似度は 48.4 と低い。特に er>はコサイン類似度が-7.8 と低く、higher との関連性がほとんど見られない。これらのサブワードから元の埋め込み表現を復元することが困難であり、これらのサブワードを足し合わせた **BoS** の埋め込み表現のコサイン類似度も 36.5 と低くなっている。対して、既知語である high の higher とのコサイン類似度は 69.8 であり、どのサブワード埋め込みよりも元の意味に近い埋め込み表現となっている。

また、固有名詞に含まれるサブワードがノイズになる場合が考えられる。例として、iceberg と Bloomberg は同じサブワード berg を共有しているが、Bloomberg の意味を berg などのサブワードから類推することは困難である。

さらに、綴り誤りや表記ゆれなどがある場合には、埋め込みを計算する上で重要なサブワードが失われ、埋め込みの品質が大きく悪化する可能性がある。例として、higher の綴りが hihger などと変化した場合には、語幹の high が失われ各サブワードの埋め込み表現から元の意味を類推することが困難である。このような表層の僅かな変化に対して、サブワード埋め込みを利用する手法は影響を受けやすいと考えられる。

また、単語埋め込みを学習した後にその埋め込みを利用してサブワード埋め込みを学習する手法は学習コストとデータ効率が悪いと考えられる。出現回数の少ないサブワードのサブワード埋め込みの性能を向上させるためには十分なサイズの訓練データが必要である。また、埋め込み表現の学習は更新するパラメータ数が多く、利用するベクトルの数に比例して学習コストが増大し、モデルのサイズが増大する。さらに、単語埋め込みとサブワード埋め込みのパイプライン学習を行うため、サブワード数を増加させて高精度な埋め込み表現を計算したとしても、既知語の埋め込み表現を復元する場合には誤差が生じ性能が劣化するという問題がある。

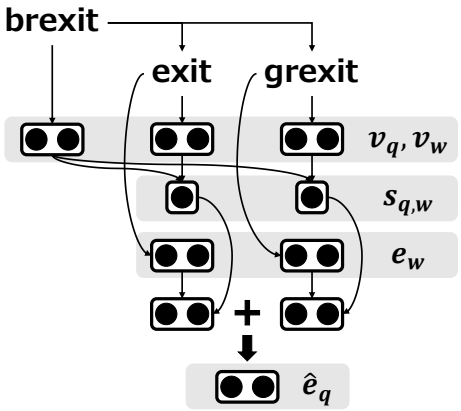

図 1　既知語の埋め込み表現を利用する手法。

次に、未知語の埋め込み表現を計算する提案手法について述べる。提案手法の概要図を図 1 に示す。本手法は、既知語 w の埋め込み表現 $\boldsymbol{e}_w$ の線形和によって未知語 q の埋め込み表現 $\hat{\boldsymbol{e}}_q$ を計算する。

$$\hat{\boldsymbol{e}}_q = \sum_i \alpha_w \boldsymbol{e}_w \tag{21}$$

このように既知語の単語埋め込みを用いることで、サブワード埋め込みを別途学習する必要が無く、大規模なテキストコーパスから学習された高性能な単語埋め込みを直接利用することが可能となる。

また、各既知語の埋め込みを足し合わせる際の重みは既知語との表層の類似性に基づいて計算する。これは表層の近い単語はその意味が近く、単語埋め込みが近くなるという仮定に基づいている。さらに、表層の類似性をニューラルネットワークを通して学習し、既知語埋め込みを計算することによって、現実的な語彙において表層の類似性と単語埋め込みの関係性を学習させる。

このようなネットワークを通して類似性の計算を未知語と全ての既知語との間に対して行うことは計算コストが大きい。そこで、本手法ではモデルを用いない高速な手法で表層の類似する既知語を抽出し、その後モデルを用いて類似性を計算する 2 段階の構成を採用した。この coarse-to-fine の構成を利用することで、既知語の語彙サイズが増加してもネットワークの計算コストは増加せず、効率的な計算が可能になる。

さらに、本手法はモデルの学習時と推論時の語彙が異なっていても適用可能である。

サブワード埋め込みを学習するモデルでは、学習時に出現しないサブワードの埋め込み表現を計算できないため、未知のサブワードを含む未知語を適切に処理することができない。そのため、単語が追加される等、語彙が変化した場合にはサブワード埋め込みを再度学習する必要がある。一方、本手法は語彙から表記ゆれ等の表層の変化と埋め込みの変化の関係を学習するため、推論時の語彙に未知語と表層の類似する単語が含まれていればその単語を手掛かりに未知語の埋め込みを計算することができる。

まず、未知語に対して表層の類似する既知語を抽出する手法として、(i) 既知語による分割 (seg) と (ii) 類似文字列検索 (approx) について述べる。

既知語による分割 未知語に含まれる既知語を抽出して利用する。まず、単語の表層を複数の既知語もしくは文字に分割する。次に、分割数が最小となる分割列に含まれる既知語を文字列が長い順に n^{seg} 個抽出する。分割数が最小となる分割列が複数存在する場合には、各分割列に含まれる既知語から同様に抽出する。単語として埋め込み表現が学習された既知語はサブワード埋め込みより意味が一義的であると期待できる。

類似文字列検索 未知語と表層の特徴量が類似する単語を既知語から抽出して利用する。まず、単語中の文字 3-gram の集合を表層の特徴量として、特徴量の Jaccard 係数を単語間の表層の類似度とする。次に、未知語に対して表層類似度の高い順に既知語を n^{approx} 個抽出する。この類似文字列検索には第 2 章で述べた CPMerge を実装した simstring [18] を用いた。

次に、上述の 2 つの方法で得られた既知語の埋め込みを利用して未知語の埋め込み表現を計算する方法について述べる。まず、未知語 q に対して、既知語による分割と類似文字列検索によって得られた既知語をそれぞれ $w_i^{\mathrm{seg}}, w_i^{\mathrm{approx}}$ とする。この既知語と未知語の表層ベクトル $\boldsymbol{v}_w$ を文字 CNN を用いて計算する。次に、$\boldsymbol{v}_w$ から未知語 q と既知語 w の間の表層の類似度 $s_{q,w}$ を計算する。その後、$s_{q,w}$ を重みとして各既知語の埋め込み $\boldsymbol{e}_w$ を足し合わせて $\boldsymbol{e}_q^{\mathrm{seg}}, \boldsymbol{e}_q^{\mathrm{approx}}$ を計算する。ここで $\boldsymbol{W}_k$ $(k \in \{\mathrm{seg}, \mathrm{approx}\})$

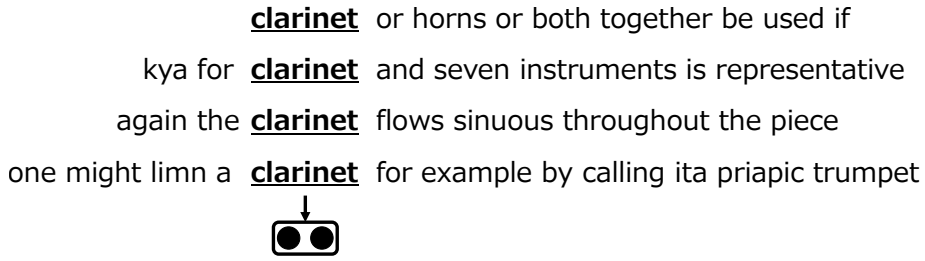

図 2　複数の文脈から未知語の埋め込み表現を計算するモデル。

は学習により推定するパラメータである。

$$s_{q,w_i^k} = \mathrm{softmax}\left(\boldsymbol{v}_q^\mathsf{T} \cdot \boldsymbol{W}_k \cdot \boldsymbol{v}_{w_i^k}\right) \tag{22}$$

$$\boldsymbol{e}_q^k = \sum_i^{n^k} s_{q,w_i^k} \boldsymbol{e}_{w_i^k} \tag{23}$$

最後に、$\boldsymbol{e}_q^{\mathrm{seg}}, \boldsymbol{e}_q^{\mathrm{approx}}$ の重み付き線形和を計算して未知語 q の埋め込み表現 $\hat{\boldsymbol{e}}_q$ とする。ここで $\boldsymbol{\theta}_k$ は学習により推定するパラメータである。

$$\alpha_k = \mathrm{softmax}\left(\boldsymbol{\theta}_k \cdot \boldsymbol{s}_{q,w_i^k}\right) \tag{24}$$

$$\hat{\boldsymbol{e}}_q = \alpha_{\mathrm{seg}} \boldsymbol{e}_q^{\mathrm{seg}} + \alpha_{\mathrm{approx}} \boldsymbol{e}_q^{\mathrm{approx}} \tag{25}$$

上述のように計算した埋め込みと既知語の埋め込みのコサイン類似度を損失関数としてモデルを訓練する。

4.2　表層の類似する単語の文脈を利用する手法

本節では文脈の情報から未知語の埋め込み表現を計算する手法について説明する。未知語の埋め込み表現を計算する際に文脈を利用するモデル [5] では、図 2 のように未知語が出現する複数の文脈を手がかりとして未知語の埋め込み表現を計算する。例えば、図において未知語 `clarinet` の意味をその単語が出現する周囲の文脈に含まれる `horn` や `instruments` といった関連する既知語から推定する。このように未知語が出現する複数の文脈を利用することで、未知語の意味を類推できるような情報が単一の文脈に含まれていない場合であっても、より高精度な埋め込み表現を推定することができる。

しかし、未知語に表記ゆれや綴り誤りが存在する場合には、表層の完全一致による検索では文脈を抽出することができないという問題がある。例えば、単語 `air-pods` が未知語として出現した場合に、`Airpods` や `AirPod` などの比較的出現頻度の高い単語の文脈を利用することができない。

この問題に対処するために、表層の類似性を学習する上述のモデルを利用する手法を提案する。例えば、単語 `air-pods` の文脈として、表層の類似する `Airpods` や `AirPod` などの単語の文脈を追加することで参照できる文脈の数を増やす。しかし、[5] の Transformer のエンコーダを利用するモデルは計算コストが高く、多数の文脈を扱うには不向きである。そのため、文脈の有用性スコアを [36] の線形モデルと [5] の Transformer モデルの 2 段階に分けて計算し、このスコアを用いて文脈の数を枝刈りする。

以下に、文脈を用いる提案手法について詳細に説明する。まず、既知語の埋め込み表現を利用する手法と同様に、類似文字列検索を用いてコーパスに出現する単語から未知語 q と表層の類似する単語 w_i を抽出する。次に、抽出した単語と未知語に対して、文字 CNN を用いて表層から表層ベクトル $\boldsymbol{v}_w$ を計算する。最後に、表層の類似スコア $s^{\text{surface}}_{q,w_i}$ を計算する。

$$s^{\text{surface}}_{q,w_i} = \text{softmax}\left(\boldsymbol{v}_q^{\mathsf{T}} \cdot \boldsymbol{W} \cdot \boldsymbol{v}_{w_i}\right) \tag{26}$$

次に、文脈を枝刈りしながら未知語の埋め込み表現を計算する手法を説明する。まず、未知語が出現する文脈と未知語と表層の類似する単語が出現する文脈を抽出する。次に、文脈に含まれる単語の埋め込み表現を基に線形モデルから文脈の有用性スコア $s^{\text{linear}}_{C_t}$ を計算する。そして、その値に先の表層の類似スコア $s^{\text{surface}}_{q,t}$ を足し合わせて、文脈の有用性スコア s^{info}_{C} を計算する。ここで、文脈 C の中心単語、文脈単語をそれぞれ t, w とする。また、$\boldsymbol{W}$ は学習により更新されるパラメータである。

$$s^{\text{linear}}_{C} = \boldsymbol{W} \cdot \frac{1}{|C|} \sum_{w \in C} \boldsymbol{e}_w \tag{27}$$

$$s^{\text{info}}_{C} = s^{\text{surface}}_{q,t} + s^{\text{linear}}_{C} \tag{28}$$

次に、文脈の有用性スコアの上位 k 件の文脈のみを Transformer のエンコーダに入力して、文脈ベクトル $\boldsymbol{e}_C$ を計算する。さらに、文脈ベクトル $\boldsymbol{e}_C$ を Transformer のエン

コーダに入力して $s_C^{\text{attention}}$ を計算し、この値を s_C^{info} と足し合わせて各文脈の重みを計算する。この重みを用いて文脈ベクトル $\boldsymbol{e}_C$ を足し合わせて未知語の埋め込み表現とする。

$$\hat{\boldsymbol{e}}_q = \sum_C \alpha_C \boldsymbol{e}_C \tag{29}$$

$$\alpha_C = \text{softmax}\left(s_C^{\text{info}} + s_C^{\text{attention}}\right) \tag{30}$$

以上に説明した提案手法と既存手法を以降の章において比較する。内的評価において表層の情報を用いる手法を比較し、外的評価において表層と文脈の情報を用いる手法をそれぞれ比較する。

第 5 章

内的評価

本章では未知語の埋め込み表現を評価するため行った内的評価について説明する。内的評価として低頻度語の類似度判定タスクと綴り誤りの埋め込み表現の評価タスクを通して未知語の埋め込み表現の性能を評価した。以下で実験設定、比較手法、実験結果、考察について順に述べる。

5.1 実験設定

内的評価に用いたデータセットと実験設定について説明する。まず、低頻度語類似度判定タスクには CARD-660 データセット [3] (CARD) を用いた。CARD はコンピュータサイエンスやソーシャルメディア、生体医学などのドメインから収集された 1306 個の単語からなる 660 組の単語対から構成されており、各単語対に人手で類似度が付与されている。同種の低頻度語の類似度判定データセットである RW [46] と比較して、注釈者間一致率が高く、**GloVe** や **fastText** などの学習みの単語埋め込みの語彙に対して未知語が多く含まれている。この類似度と単語対の埋め込み表現のコサイン類似度との相関をスピアマンの順位相関係数で評価した。なお、未知語の埋め込み表現を評価するために、既知語に関しては学習済みの埋め込み表現を用い、未知語については各手法で埋め込み表現を計算して評価を行った。先行研究 [3] と同様に全ての単語対についての評価と、未知語を含む単語対に限定した評価を行った。

次に、綴り誤りに対する単語埋め込みの頑健性を評価するために、英語学習者の作成したエッセイのコーパス [4] (TOEFL) を利用して評価を行った。具体的には、TOEFL データセットから綴りの正しい単語 (既知語) と綴りの誤った単語 (未知語) の

組を 1514 組抽出し、これを用いて評価を行った。埋め込み表現を評価に利用するために綴りの正しい単語の中から既知であるものを利用し、綴り誤りが高頻度であって既に学習された埋め込み表現を利用できる状況を避けるために綴りの誤った単語の中から未知であるものを利用した。各手法を用いて綴り誤りの単語の埋め込み表現を計算し、正しい綴りの単語の埋め込み表現とのコサイン類似度を評価した。

なお、内的評価において各手法が未知語の埋め込み表現を計算できない場合には、既存研究 [45] に倣ってその単語を含む単語対の類似度を 0 として計算した。

5.2　比較手法

以下に述べる手法について比較実験を行った。

GloVe [7] 学習済みの埋め込み表現[*2]を利用して既知語の埋め込み表現に利用し、未知語を含む単語対については類似度を 0 とする。

fastText [9] 学習済みのサブワード埋め込み[*3]を用いて、未知語を含む全ての単語の単語埋め込みを計算する。

BoS [1] **GloVe** の語彙に含まれる n-gram に対してサブワード埋め込みを学習しておき、未知語の埋め込み表現は未知語に含まれるサブワードの埋め込み表現の平均として計算する。語彙に含まれる長さ 3-6 の文字 n-gram を全て対象として、それらのサブワード埋め込みを学習する。約 200 万単語からなる語彙から抽出されるサブワードの数は約 620 万である。公開されているコード[*4]を利用してモデルを訓練した。

SUM-F [2] **BoS** と同様にサブワード単位の埋め込み表現を学習して用いるが、埋め込みを計算するサブワードの長さが 3-30 である点と、出現頻度が上位 F 件のサブワードのみに限定する点が異なる。ハイパーパラメータは $F = 0.5\text{M}$ とした。

[*2] https://nlp.stanford.edu/projects/glove
[*3] https://fasttext.cc/docs/en/english-vectors.html
[*4] https://github.com/jmzhao/bag-of-substring-embedder

公開されているコード[*5]を利用してモデルを訓練した。

KVQ-FH [2] サブワード埋め込みを計算する手法であり、埋め込み表現を計算するサブワードを出現頻度が上位 F 件のものに限定し、ハッシュ関数を用いてサイズ H の埋め込み表現を異なるサブワード間で共有し、埋め込み表現の集約時に Self-attention を計算する。ハイパーパラメータとして $F = 1\text{M}, H = 0.5\text{M}$ を用いた。これらの値は大きいほど元の単語埋め込みの復元性能が向上し、また未知語の埋め込み表現の性能も向上する [2] が、同時に学習コストやパラメータサイズが増大するトレードオフの関係にある。公開されているコード[*4]を利用してモデルを訓練した。

提案手法（表層） 提案手法のハイパーパラメータは $n^{\text{seg}} = 10, n^{\text{approx}} = 10$ とした。また、表層の類似度を計算するための文字埋め込みの次元は 100 とし、文字 CNN のフィルタサイズは 2,4,6,8 とした。学習済みの単語埋め込みの既知語の内、頻度が 10^3 から 10^5 の単語の埋め込み表現を用いてモデルを訓練した。単語埋め込みの語彙の頻度には [2] と同じものを利用した。訓練データからランダムにサンプルした 1000 例を開発データとして、開発データでの評価が最も良かったパラメータを評価に用いた。損失関数の最適化には Adam [47] を用いた。実験に用いたその他のハイパーパラメータを表 2 に示す。PyTorch ver 1.0[*6]を用いて実装した。

各比較手法の学習には学習済みの単語埋め込みを用いた。学習済みの単語埋め込みには **GloVe** [7] と **fastText** [9] を用いた。

5.3　実験結果

内的評価の実験結果について述べる。まず、内的評価の実験結果を表 3、表 4 に示す。表中の CARD における ALL は全ての単語対での評価であり、OOV は未知語を含む単語対に限定した評価である。**GloVe** では、未知語に対して単語埋め込みを計算

[*5] https://github.com/losyer/compact_reconstruction
[*6] https://pytorch.org/

学習率	10^{-3}
バッチサイズ	10^3
最大エポック数	50
勾配クリッピング	1.0

表 2　実験に用いた提案手法のハイパーパラメータ。

	CARD	
	ALL	OOV
GloVe [7]	27.3	-
BoS [1]	37.3	18.0
SUM-F [2]	41.9	21.7
KVQ-FH [2]	45.5	28.8
提案手法	**48.3**	**37.1**

	TOEFL
GloVe [7]	-
BoS [1]	11.6
SUM-F [2]	11.6
KVQ-FH [2]	10.9
提案手法	**36.2**

表 3　未知語の埋め込み計算手法の内的評価（単語埋め込み：**GloVe**）。

	CARD	
	ALL	OOV
fastText [9]	42.4	11.7
BoS [1]	51.7	30.3
SUM-F [2]	52.0	34.3
KVQ-FH [2]	49.9	29.6
提案手法	**55.0**	**38.9**

	TOEFL
fastText [9]	39.8
BoS [1]	54.3
SUM-F [2]	45.1
KVQ-FH [2]	43.4
提案手法	**69.6**

表 4　未知語の埋め込み計算手法の内的評価（単語埋め込み：**fastText**）。

できないため、CARD の OOV と TOEFL の結果をそれぞれ評価無しとした。一方、**fastText** ではサブワード埋め込みを用いて未知語にも単語埋め込みを計算することができるが、その性能は他の比較手法に比べて低い。これは既存研究 [2] と同様の傾向であり、既知語の埋め込み表現を用いた再構成タスクが未知語の埋め込み表現の計算が

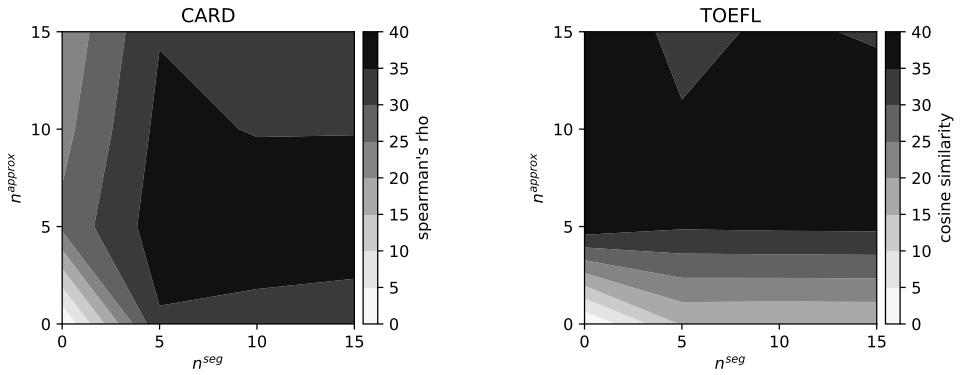

図 3　感度分析の結果。

有効であることを示唆している。また、どちらのデータセットにおいても既存手法と比較して提案手法の性能が向上した。特に、提案手法によって綴り誤りにより頑健な単語埋め込みが計算できることが示された。このことから、サブワード埋め込みが綴り誤りに対して影響を受けやすいことと、提案手法によってこの問題に対処できることが示唆された。

5.4　考察

本節では、内的評価の考察について説明する。以下で、提案手法の感度分析、単語のタイプ別の CARD の分析、埋め込み表現の精度と単語の長さ・頻度の関係、他言語での分析、埋め込み表現の出力例について順に説明する。

5.4.1　提案手法の感度分析

次に、提案手法のハイパーパラメータの感度分析について述べる。提案手法の既知語による分割と類似文字列検索において、それぞれ $n^{\mathrm{seg}}, n^{\mathrm{approx}} \in \{0, 5, 10, 15\}$ を変化させて、事前学習した単語埋め込みに **GloVe** を用いた場合の CARD の未知語を含む単語対 (表 3 の OOV) と TOEFL で評価した結果を図 3 に示す。各図において、横軸は n^{seg}、縦軸は n^{approx} であり、図の濃淡はそれぞれスピアマンの相関係数とコサイン

	ALL	固有名詞	固有名詞以外
BoS	25.2	31.4	23.9
KVQ-FH	33.5	37.9	31.5
提案手法	38.6	58.6	33.5

表 5　単語のタイプ別の CARD の結果。

類似度の平均を表す。各図の原点 $n^{\text{seg}} = 0, n^{\text{approx}} = 0$ では各指標の値を 0 として表示した。図 3 の CARD の結果から既知語による分割が CARD に有効であることが示された。これは CARD に含まれる合成語に対して既知語による分割が効果的であるためと考えられる。また、図 3 の TOEFL の結果から類似文字列検索を利用することによって綴り誤りに対して頑健になることが示された。既知語による分割で抽出する単語数が多いほどより短い単語を考慮でき、類似文字列検索で抽出する単語数が多いほどより表層の離れた単語を考慮できる。しかし、参照する単語数が多いとモデルの学習・推論コストを増大させるため、これらはトレードオフの関係にあると考えられる。

5.4.2　単語のタイプ別の CARD の分析

次に、内的評価における低頻度語を分類して各手法の性能を比較した。単語埋め込みが **GloVe** の場合の各手法の性能を CARD のデータにおいて未知語のタイプ別に分析した結果を表 5 に示す。まず、CARD のデータに含まれる単語対について、片方のみの単語が **GloVe** の未知語である 205 組を抽出し、各単語対に含まれる未知語が固有名詞と固有名詞以外とに単語対を分類した。未知語を分類する際には、その単語に綴り誤りや表記ゆれがある場合にも、その単語が指すエンティティが固有名詞であるかどうかについて分類した。固有名詞の判別には BabelNet[*7]の word-type (Named Entity, Concept) を利用し、登録されていない未知語については人手で注釈付けを行った。次に、各手法によって未知語の埋め込み表現を計算し、既知語の埋め込みとのコサイン類似度を評価した。結果としてサブワード埋め込みを利用する手法と比較して、提案手法によって固有名詞の性能が向上した。このことから、形態素と意味の関係が薄い固

[*7] https://babelnet.org/

有名詞などの未知語には既知語の埋め込み表現を直接利用する手法が有効であることが示された。

5.4.3　埋め込み表現の精度と単語の長さ・頻度の関係

次に、各手法によって計算される未知語の埋め込み表現の精度と、未知語の長さや出現頻度との関係を調べた。比較手法として、**SUM-F**、**KVQ-FH**、提案手法を用い、各手法のハイパーパラメータは内的評価と同じものを利用した。

学習済みの単語埋め込みの頻度上位 50 万件を学習データとして各手法を学習し、残りの出現頻度の低い単語を未知語として埋め込み表現を計算し、コサイン類似度で評価を行った。単語埋め込みには **GloVe** を用い、単語の出現頻度は [2] と同じものを利用した。単語の出現頻度とコサイン類似度の関係を図 4 に、単語の長さとコサイン類似度の関係を図 5 に示す。各図における縦軸はコサイン類似度の平均値である。

図 4 から、短い単語について提案手法による埋め込み表現の推定精度が高く、単語が長くなるにつれてサブワード埋め込みを用いる手法の精度が向上した。このことから、長い単語には合成的に作られた単語が含まれておりサブワードから単語を再構成することで埋め込み表現を推定できていると考えられ、短い単語ではサブワード数が少ないためノイズとなるサブワードの影響が大きくなり精度が低下していると考えられる。図 5 から、単語の頻度と推定精度について各手法で傾向の違いは見らなかった。図において、出現頻度が増加するほど各手法の推定精度が増加しているが、特に出現頻度の低い単語の推定精度が高くなっている。これは、日付や金額などの数字を含むような単語による影響であると考えられる。

5.4.4　他言語での分析

次に、英語以外の言語について行った実験について説明する。まず、サブワード埋め込みを用いる手法 (**SUM-F, KVQ-FH**) と本手法の計算した単語埋め込みの精度を品詞別に分析した結果について説明する。まず、単語埋め込みの評価データとして品詞タグ付けのデータセットに出現する単語を抽出した。次に、単語埋め込みの語彙から評価データを除いた単語の埋め込み表現を用いて各手法を訓練した。最後に、各手法で評価データの単語の埋め込み表現を計算し、正しい埋め込み表現とのコサイン類似度を

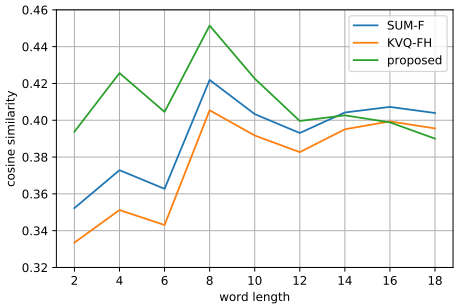

図 4　単語の長さとコサイン類似度の関係。

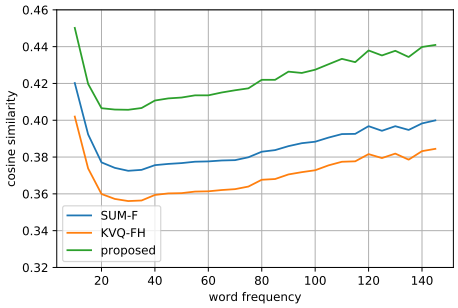

図 5　単語の出現頻度とコサイン類似度の関係。

	英語	日本語	トルコ語	スペイン語
SUM-F	58.1	29.2	67.3	65.3
KVQ-FH	57.9	33.4	68.5	64.8
提案手法	64.4	48.2	71.0	68.8

表 6　各言語における埋め込み表現の精度。

計算し評価した。また、品詞に Universal POS を用い、内容語である名詞 (NOUN)、固有名詞 (PROPN)、動詞 (VERB)、形容詞 (ADJ)、副詞 (ADV)、間投詞 (INTJ)、句読点 (PUNCT)、記号 (SYM)、その他 (X) について分析を行った。

実験には英語、日本語、トルコ語、スペイン語を用いた。英語と比較して、トルコ語は単語を合成語の多い言語であり、スペイン語は動詞等の活用の多い言語である。各言語の単語埋め込みには fastText [9][*8]を用いた。また、品詞タグ付けのデータセットには Universal Dependencies v2.4 から、英語は English-EWT、日本語は Japanese-BCCWJ、トルコ語は Turkish-IMST、スペイン語は Spanish-AnCora を用いた。

実験結果を表 6、図 6、図 7、図 8、図 9 に示す。まず、表 6 において、トルコ語の精度が他の 2 つよりも高く、トルコ語が表層の情報から意味を類推しやすいことが示唆さ

[*8] https://fasttext.cc/docs/en/crawl-vectors.html

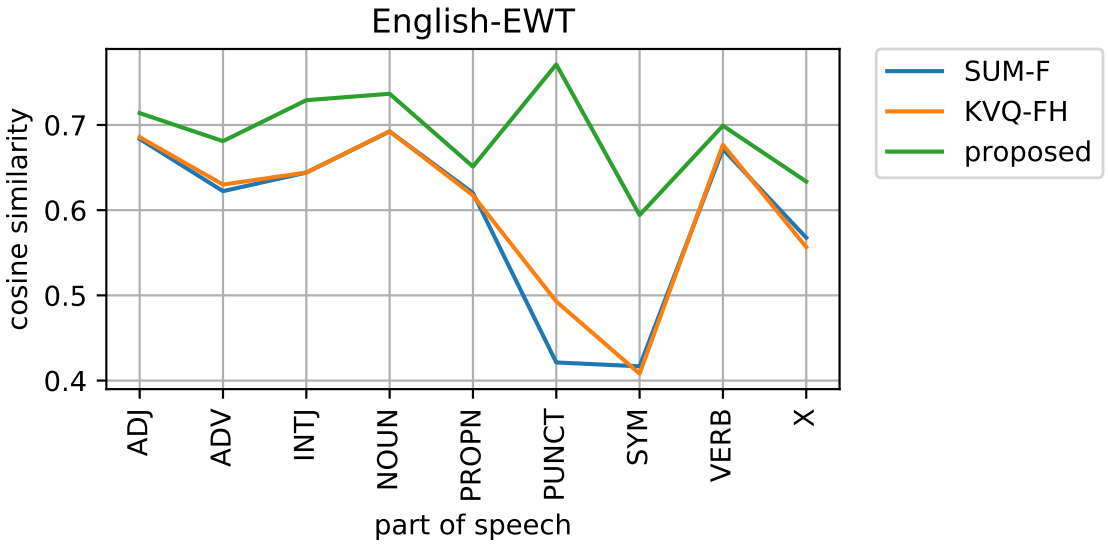

図 6　品詞毎の埋め込み表現の精度（英語）。

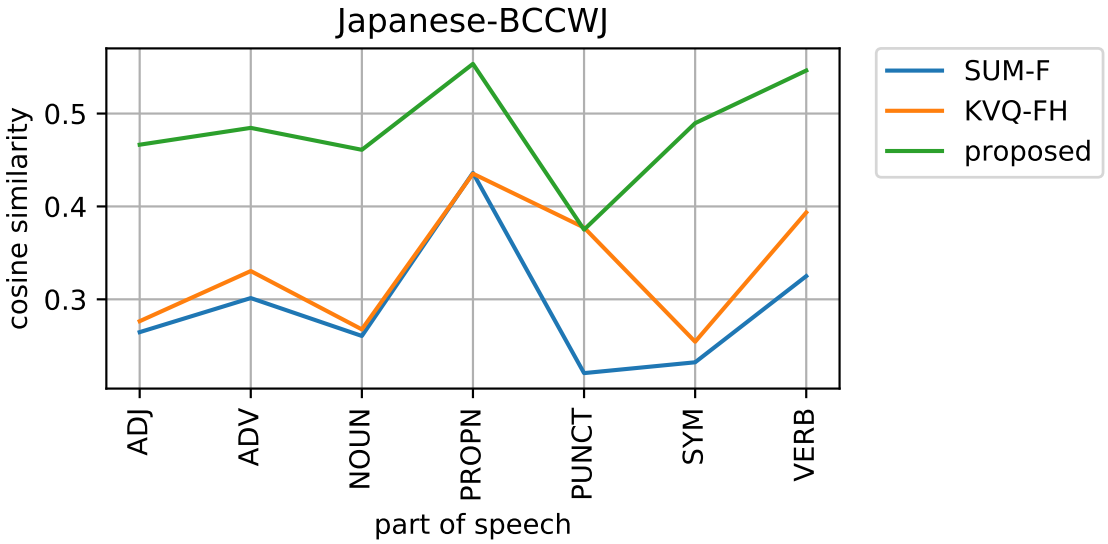

図 7　品詞毎の埋め込み表現の精度（日本語）。

れる。これは既存研究 [1] でも言及されており、合成語の多い言語にはサブワードから単語埋め込みを推定する手法が効果的であることが示唆される。次に、英語に比べて他

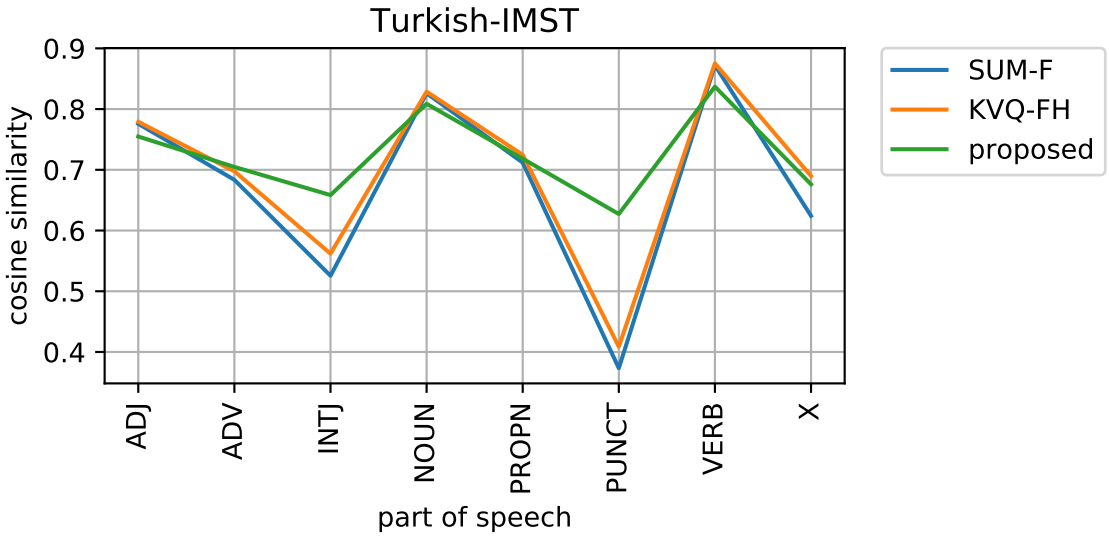

図 8　品詞毎の埋め込み表現の精度（トルコ語）。

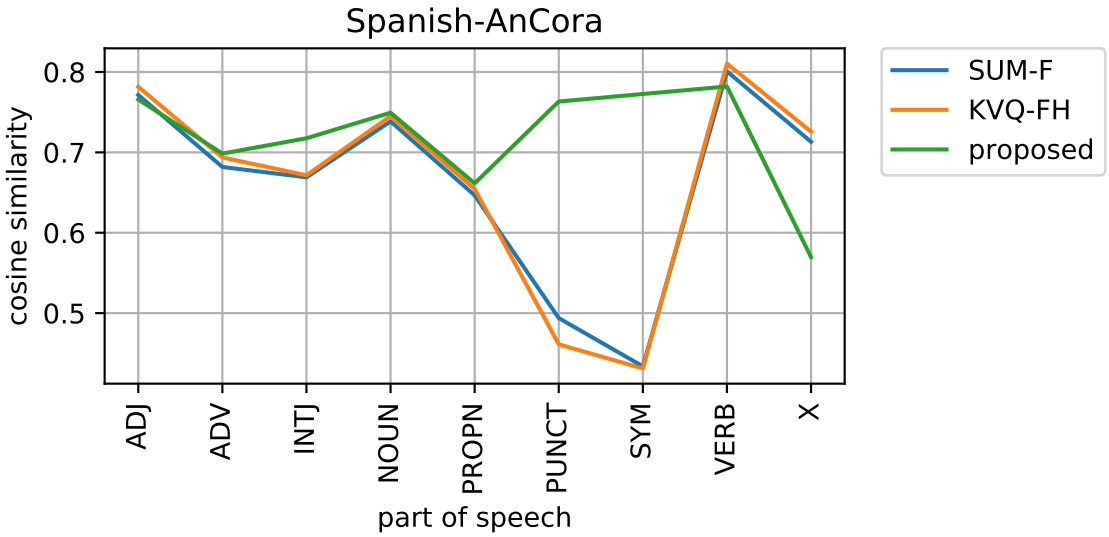

図 9　品詞毎の埋め込み表現の精度（スペイン語）。

の二つの言語では提案手法と比較手法の精度の差が小さく（表 6）、特に動詞 (VERB) において比較手法の精度が高い（図 8、図 9）。このことから、規則的な動詞の活用に

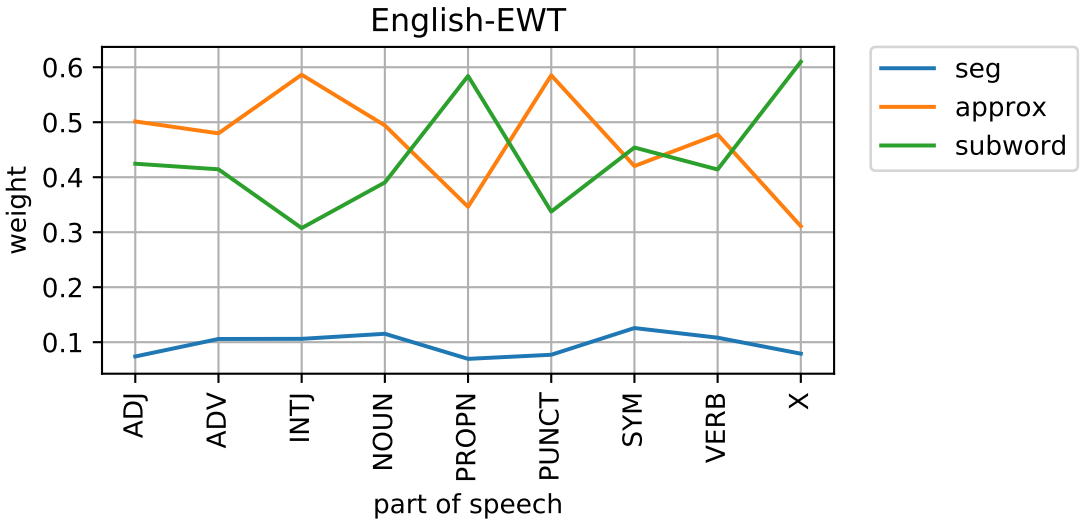

図 10　seg, approx, subword の重みと品詞の関係（英語）。

はサブワード埋め込みが効果的であることが示唆される。また、句読点 (PUNCT) において提案手法の精度が高いが、これは PUNCT に . . . や ?! といった記号の順番や数によって意味が変化しない単語が含まれており、表記ゆれに頑健な提案手法が効果的であるためと考えられる。

次に、提案手法の既知語による分割から計算した埋め込み表現と、類似文字列検索によって計算した埋め込み表現に、サブワード埋め込みを利用した埋め込み表現を加えた、3 つの埋め込み表現の寄与度の分析を単語の品詞別に行った。

まず、各手法によって計算される埋め込み表現を組み合わせる手法について説明する。既知語による分割 (seg) と類似文字列検索 (approx) と SUM-F [2] (subword) によって計算された埋め込み表現を以下の式のように足し合わせる。ここで、$\boldsymbol{e}_q^{\text{seg}}, \boldsymbol{e}_q^{\text{approx}}, \boldsymbol{e}_q^{\text{subword}}$ はそれぞれ各手法によって計算された埋め込み表現であり、$\boldsymbol{W}$ は学習によって計算されるパラメータである。また、$[;]$ は結合操作を表す。

$$\hat{\boldsymbol{e}}_q = \sum_k \alpha_k \boldsymbol{e}_q^k \ (k \in \{\text{seg}, \text{approx}, \text{subword}\}) \tag{31}$$

$$\alpha_k = \text{softmax}(\boldsymbol{W} \cdot [\boldsymbol{e}_q^{\text{seg}}; \boldsymbol{e}_q^{\text{approx}}; \boldsymbol{e}_q^{\text{subword}}]) \tag{32}$$

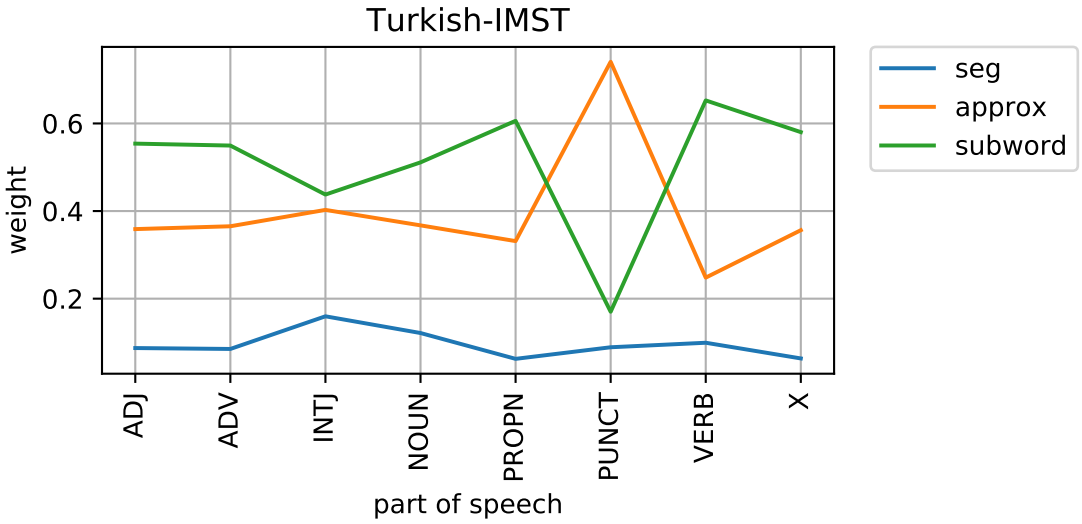

図 11　seg, approx, subword の重みと品詞の関係（トルコ語）。

このように計算した単語埋め込みを、既知語の埋め込み表現を用いて訓練した。次に、訓練したモデルを用いて、品詞タグ付けのデータセットに出現する単語の埋め込み表現を計算する時の各埋め込み表現 $\boldsymbol{e}_q^{\text{seg}}, \boldsymbol{e}_q^{\text{approx}}, \boldsymbol{e}_q^{\text{subword}}$ の重み $\alpha_{\text{seg}}, \alpha_{\text{approx}}, \alpha_{\text{subword}}$ を計算した。

実験の結果を図 10、図 11 に示す。横軸は品詞であり、縦軸は $\alpha_{\text{seg}}, \alpha_{\text{approx}}, \alpha_{\text{subword}}$ を各品詞について平均した値である。まず、どの言語においても、seg の重みが他の二つと比較して低い。このことから、seg の役割が subword と重複しており、subword と組み合わせた時にはその重要性が低下することが示唆される。次に、トルコ語ではほぼ全ての品詞において subword の重みが高く、先の実験と同じ傾向が見られた。また、どの言語においても固有名詞 (PROPN) において approx より subword の重みが大きくなっており、先の実験と傾向が異なる。これは、サブワード埋め込みはパラメータ数が多く表現能力が高く、過学習しやすいためであると考えられる。また、英語で見られない傾向として、トルコ語とスペイン語の動詞において、subword の重みが大きくなっている。最後に、句読点 (PUNCT) において approx の重みが大きいが、これは先の実験と同じ傾向であり、approx が表記ゆれに頑健であることの効果が現れていると考えられる。

単語	手法	類似度	埋め込み表現の近傍の既知語				
blessings	GloVe	100	blessings	blessing	blessed	Blessings	bless
	BoS	50	doesnamounts	commonains	deliverances	exaltations	chastisements
	提案手法	85	blessing	blessings	Blessings	blessed	bless
espically	GloVe	100	espically	exspecially	especally	espeically	espescially
	BoS	48	budgetarily	similarily	charily	particualrly	palatably
	提案手法	77	especally	espescially	esecially	escpecially	epsecially
clumsy	GloVe	100	clumsy	awkward	clunky	ungainly	inept
	BoS	71	clumbsy	clumsy	klutzy	awkward-looking	ungainly
	提案手法	37	clumsey	clumbsy	clumpsy	stumbly	hamhanded
LANs	GloVe	100	LANs	WANs	WLANs	VPNs	SANs
	BoS	78	LANs	WANs	WLANs	MANs	VLANs
	提案手法	54	LAN	WAN	WLAN	ETHERNET	LANs

表 7　提案手法の出力例。

5.4.5　埋め込み表現の出力例

次に、計算された埋め込みの出力例の分析について述べる。提案手法によって計算された埋め込み表現の近傍の単語を **BoS** と比較した。**GloVe** の既知語の表層を基に、**BoS** と提案手法によって計算した埋め込み表現の元の埋め込み表現とのコサイン類似度と、計算した埋め込み表現の近傍上位 5 個の既知語を表 7 に示す。上 2 つの例は提案手法が **BoS** より元の単語に近い埋め込み表現を計算した例であり、下 2 つの例は提案手法の計算した埋め込みが元の埋め込みから離れている例である。1 つ目の `blessing` の複数形である `blessings` の例や、2 つ目の `especially` の綴り誤りである `espically` の例において、提案手法の計算した埋め込み表現の元の埋め込み表現に対する類似度が高く、また近傍語も **GloVe** のものと類似した結果となっている。一方、これらの単語に対して **BoS** では比較的元の埋め込みから離れた埋め込みを出力しており、`blessings` や `espically` の接尾辞 `s` や `ly` に強く影響を受けていることが窺がえる。対して、3 つ目の `clumsy` や 4 つ目の `LANs` の例においては、類似度の傾向が逆転している。3 つ目の `clumsy` の例では、提案手法の計算した埋め込み表現の近傍に `clumsy` の綴り誤りである `clumsey` や `clumbsy` が見られることから、綴り誤りの単語を参照したことによる悪影響を受けていると考えられる。また、4 つ目の `LANs` の

例では、提案手法の計算した埋め込み表現の近傍に LAN が見られることから、提案手法が学習した表層類似性において接尾辞の s が単語の意味を大きく変えないと学習した結果が現れていると考えられる。このような単数形と複数形の混同は品詞タグ付け等の構文的情報を利用するタスクにおいて悪影響を及ぼすと考えられる。

第 6 章

外的評価

本章では未知語の埋め込み表現を評価するため行った外的評価について説明する。外的評価として未知語の埋め込み表現の品質の下流タスクへの影響を品詞タグ付けと固有表現抽出を通して評価した。以下で、実験設定、比較手法、実験結果、考察について順に述べる。

6.1 実験設定

外的評価で用いたデータセットと実験設定について説明する。未知語が多く含まれるデータで外的評価を行うために、データセットにはツイッターと生物医学のドメインのものを用いた。外的評価に用いたデータセットの情報を表 8 に示す。表中の OOV はデータセットの語彙に占める **GloVe** に対する未知語が占める割合である。また、POS は品詞タグ付けを、NER は固有表現抽出を指す。

ツイッタードメインの品詞タグ付けにはツイートに品詞が付与された ARK [48]、T-POS [49]、DCU [50] データセットを用いた。ただし、DCU の訓練データには T-POS と同じものを用いた。品詞タグ付けのモデルには LSTM モデル [26] を利用し、単語単位の分類精度を評価した。

ツイッタードメインの固有表現抽出にはツイートに固有表現が付与された Rare-NER [51]、及び Multi-NER データセット [52] を用いた。生物医学ドメインの固有表現抽出には生物医学に関する固有表現が付与された BC2GM（遺伝子・たんぱく質）[53]、BC4CHEMD（化学式）[54]、BC5CDR（化学式・疾患）[55]、NCBI-Disease（疾患） [56] データセットを用いた。固有表現抽出のモデルには LSTM-CRF モデル [57]

	タスク	ドメイン	文	OOV
ARK [48]	POS	ツイッター	1827	0.29
T-POS [49]	POS	ツイッター	787	0.20
DCU [50]	POS	ツイッター	519	0.10
Rare-Ner [51]	NER	ツイッター	5690	0.27
Multi-NER [52]	NER	ツイッター	8257	0.49
BC2GM [53]	NER	生物医学	20131	0.22
BC4CHEMD [54]	NER	生物医学	87685	0.29
BC5CDR [55]	NER	生物医学	13938	0.08
NCBI-Disease [56]	NER	生物医学	7287	0.13

表 8　外的評価の実験に用いたデータセット。

LSTM 中間層の次元	200
LSTM 層数	2
ドロップアウト	0.5
学習率	10^{-3}
最大エポック数	50
勾配クリッピング	1.0

表 9　外的評価の実験に用いたハイパーパラメータ。

を利用し、エンティティ単位の F1 値を評価した。また、固有表現抽出のタグのフォーマットを IOB2 に統一して実験を行った。

単語埋め込みの語彙に対する未知語が、訓練データに含まれている場合がある。その場合には、その未知語の埋め込み表現の精度が低くとも、モデルが訓練データを用いてラベルとの関係を学習することができる。このような学習を防ぐため、訓練データからテストデータに出現する未知語を含む文を除いて実験を行った。さらに、タスクの LSTM モデル、LSTM-CRF モデルの LSTM をそれぞれ Feed forward neural network (FFN) に置き換えたモデルでも実験を行った。以下、LSTM を用いるモデルを w/ LSTM、FFN を用いるモデルを w/o LSTM と呼称する。このようにモデルを

より簡素にすることで、訓練データからタスク特有の単語埋め込みを訓練することを防ぎ、より直接的に単語埋め込みの性能を比較することができると考えられる。

各モデルの学習には Adam [47] を用いた。また、各データセットの開発データにおいて最も精度の良いパラメータを用いて評価を行った。タスクの学習時のバッチサイズはツイッタードメインでは 50 とし、生物医学ドメインでは 500 とした。その他の実験に用いた各モデルのハイパーパラメータを表 9 に示す。

6.2　比較手法

次に、比較手法について説明する。外的評価では、表層の情報を用いる手法に加えて、文脈の情報を用いるモデルの評価も行う。内的評価で述べた手法に加えて以下に示すベースラインとの比較を行った。

Single-UNK 未知語に単一の未知語ベクトル割り当て、タスクと同時に訓練する。

HiCE（文脈のみ） [5] Transformer のエンコーダブロックを用いて複数の文脈をエンコードし、未知語の埋め込み表現を計算する。文脈のウィンドウ幅を 10、最大文脈数を 10 とし、他のハイパーパラメータは公開されているコード[*9]と同じ値を利用した。なお、文脈のモデルのみを利用する点が [5] と異なる。

提案手法（文脈） 表層の類似性を計算するモデルは内的評価と同様である。利用する文脈の数を 100 とし、枝刈りする際の基準は上位 10 件までとした。

各手法の訓練データには、埋め込み表現が存在し、外部テキストコーパスに出現する単語を訓練データとして利用した。特に、文脈の情報を用いるモデルの訓練には、外部テキストコーパスにおいて単語が出現する文を文脈として利用した。またテスト時には、外部テキストコーパスと、テストデータを含むタスクのテキストデータから、未知語が出現する文脈を抽出して未知語の埋め込み表現を計算した。[5] と同様に外部テキストコーパスには Wikitext-103 [58] を用いた。単語埋め込みの性能を比較するために、[1] に倣って、各手法によって計算した単語埋め込みを固定してタスクのモデルを

[*9] https://github.com/acbull/HiCE

	表層	文脈	**ARK**		**T-POS**		**DCU**		**Rare**		**Multi**	
			ALL	OOV	ALL	OOV	ALL	OOV	ALL	OOV	ALL	OOV
Single-UNK			82.9	53.8	81.1	56.7	82.2	60.8	37.5	5.1	**69.2**	27.3
BoS	✓		82.6	52.7	80.4	55.2	81.8	62.1	35.3	5.1	68.4	27.9
KVQ-FH	✓		82.7	53.4	80.1	54.1	81.8	61.3	37.6	7.1	68.5	27.4
提案手法 (表層)	✓		**85.0**	**74.3**	**81.6**	**67.8**	**82.3**	**73.7**	37.3	9.3	68.9	**38.5**
HiCE(文脈)		✓	81.0	53.9	80.7	55.4	81.2	63.4	37.0	3.4	67.5	27.4
提案手法 (文脈)		✓	81.9	58.6	80.9	60.1	81.7	65.3	37.5	5.1	67.3	32.1
提案手法 (表層 + 文脈)	✓	✓	84.4	68.9	81.3	64.0	82.0	70.4	**38.0**	**12.9**	67.3	35.4

	表層	文脈	**BC2**		**BC4**		**BC5**		**NCBI**	
			ALL	OOV	ALL	OOV	ALL	OOV	ALL	OOV
Single-UNK			77.5	76.7	85.0	71.6	84.8	64.2	81.8	64.4
BoS	✓		77.9	76.3	84.9	70.7	84.9	62.6	81.5	68.1
KVQ-FH	✓		77.8	76.3	85.0	71.7	85.1	62.0	81.7	66.9
提案手法 (表層)	✓		**78.6**	**79.4**	**86.7**	**79.9**	85.4	**74.9**	83.0	**76.6**
HiCE(文脈)		✓	77.0	75.8	84.9	72.4	84.4	62.4	**83.2**	68.3
提案手法 (文脈)		✓	77.7	75.7	85.5	74.7	84.7	64.3	82.2	72.1
提案手法 (表層 + 文脈)	✓	✓	78.0	77.9	86.6	78.9	**85.6**	74.0	82.7	73.2

表 10　品詞タグ付けと固有表現抽出の結果の比較（単語埋め込み：**GloVe**、モデル：w/ LSTM）。

学習し、評価を行った。

6.3　実験結果

外的評価の結果を表 10 から表 13 に示す。なお、一部のデータセットの名称を Rare-NER (Rare)、Multi-NER (Multi)、BC2GM (BC2)、BC4CHEMD (BC4)、BC5CDR (BC5)、NCBI-Disease (NCBI) のように略記した。表中の ALL は全単語対での評価である、OOV は品詞タグ付けにおいては、未知語に限定して品詞を評価し

	表層	文脈	**ARK**		**T-POS**		**DCU**		**Rare**		**Multi**	
			ALL	OOV	ALL	OOV	ALL	OOV	ALL	OOV	ALL	OOV
Single-UNK			81.9	51.1	78.9	47.4	81.9	53.1	36.9	2.2	66.7	24.3
BoS	✓		82.3	50.4	79.1	47.8	81.8	55.4	**38.9**	6.4	**66.9**	26.3
KVQ-FH	✓		82.0	49.8	79.1	47.4	81.9	55.6	37.7	6.4	66.6	24.4
提案手法 (表層)	✓		**83.6**	**64.9**	80.0	**61.2**	**82.1**	**62.3**	36.2	12.7	66.5	38.3
HiCE(文脈)		✓	79.8	51.2	77.3	52.4	79.7	54.9	36.1	0.6	63.9	23.3
提案手法 (文脈)		✓	80.8	57.4	78.1	57.7	80.4	60.5	35.5	3.6	65.3	33.8
提案手法 (表層 + 文脈)	✓	✓	83.5	64.3	**80.1**	60.5	82.0	58.4	37.1	**16.1**	66.3	**40.5**

	表層	文脈	**BC2**		**BC4**		**BC5**		**NCBI**	
			ALL	OOV	ALL	OOV	ALL	OOV	ALL	OOV
Single-UNK			74.2	74.8	84.0	70.2	80.4	61.5	78.0	56.0
BoS	✓		74.0	73.4	84.3	71.8	80.5	48.7	76.9	47.0
KVQ-FH	✓		73.8	73.6	84.3	71.6	80.7	48.1	76.2	47.8
提案手法 (表層)	✓		**75.0**	**76.3**	85.3	77.3	81.3	70.4	**78.0**	**64.1**
HiCE(文脈)		✓	72.2	72.7	82.5	70.2	79.8	56.8	76.4	54.1
提案手法 (文脈)		✓	72.7	73.4	83.2	72.1	79.9	58.1	77.5	63.8
提案手法 (表層 + 文脈)	✓	✓	74.6	75.5	**85.6**	**78.4**	**81.7**	**71.3**	77.8	64.1

表 11　品詞タグ付けと固有表現抽出の結果の比較（単語埋め込み：**fastText**、モデル：w/ LSTM）。

た性能を表し、固有表現抽出においては、単語の中に未知語を含むエンティティに限定して評価した性能を表す。まず、単独の未知語埋め込みを利用する **Single-UNK** と比較して、表層や文脈を利用するモデルの OOV の性能が向上した。これは下流タスクにおいても未知語埋め込みを計算することで精度が向上することを示している。一方で、Multi-NER や Rare-NER において **Single-UNK** の ALL の性能が他の比較手法よりも高いものがある。これは、未知語を含まない多数のエンティティに対して、比較手法による未知語埋め込みがノイズとなって悪影響を及ぼしたためと考えられる。次に、表層を利用する手法の間では内的評価と同様に、既存手法と比較して提案手法

	表層	文脈	**ARK**		**T-POS**		**DCU**		**Rare**		**Multi**	
			ALL	OOV	ALL	OOV	ALL	OOV	ALL	OOV	ALL	OOV
Single-UNK			81.5	40.8	80.3	41.6	80.1	31.5	**35.4**	0.0	66.9	12.3
BoS	✓		82.3	49.9	80.2	41.6	79.5	31.5	33.6	0.0	66.6	12.4
KVQ-FH	✓		82.5	51.8	80.3	41.8	79.5	31.5	33.7	0.0	66.7	13.5
提案手法 (表層)	✓		**84.2**	**71.4**	**82.0**	**68.2**	81.3	**70.2**	34.9	5.6	**66.9**	22.3
HiCE(文脈)		✓	79.0	40.0	79.4	42.5	79.0	36.6	30.5	2.3	64.0	18.1
提案手法 (文脈)		✓	80.7	49.8	80.2	48.1	79.7	57.0	32.1	4.1	64.6	25.3
提案手法 (表層 + 文脈)	✓	✓	83.6	64.8	81.8	61.8	**81.4**	66.1	33.7	**10.5**	65.0	**26.3**

	表層	文脈	**BC2**		**BC4**		**BC5**		**NCBI**	
			ALL	OOV	ALL	OOV	ALL	OOV	ALL	OOV
Single-UNK			58.1	33.0	67.7	16.0	79.1	17.1	73.1	36.4
BoS	✓		58.1	32.5	67.8	16.6	79.5	15.7	73.6	26.8
KVQ-FH	✓		58.2	32.6	67.6	16.4	79.4	15.2	73.8	25.9
提案手法 (表層)	✓		**64.5**	63.2	70.4	41.7	80.7	56.4	74.7	63.3
HiCE(文脈)		✓	60.4	**63.3**	67.1	32.9	78.7	45.0	74.4	53.2
提案手法 (文脈)		✓	60.1	60.2	68.9	42.5	79.6	47.3	74.2	55.2
提案手法 (表層 + 文脈)	✓	✓	63.4	61.2	**70.9**	**46.2**	**80.8**	**62.3**	**75.8**	**72.1**

表 12　品詞タグ付けと固有表現抽出の結果の比較（単語埋め込み：**GloVe**、モデル：w/o LSTM）。

の OOV の精度が向上した。また、文脈を利用する手法の間では、既存手法と比較して OOV の精度が同程度もしくは向上した。これは、既存手法が参照できる文脈の数が少ないためであると考えられる。例えば、ツイッタードメインのデータセットを通して、既存手法が参照した未知語の文脈の数は中央値が 1 であった。次に、表層を利用する手法と文脈を利用する手法を比較すると、表層を利用する手法の精度が高い。文脈を考慮した単語表現を学習しない w/o LSTM においても同様の傾向が見られた。さらに、表層と文脈を組み合わせるモデルと比較しても w/ LSTM においては表層のモデルの精度が高い。LSTM を通して訓練データから文脈を考慮した単語表現を学習するため、

	表層	文脈	**ARK**		**T-POS**		**DCU**		**Rare**		**Multi**	
			ALL	OOV	ALL	OOV	ALL	OOV	ALL	OOV	ALL	OOV
Single-UNK			79.6	37.0	79.4	36.7	80.4	26.9	**38.0**	1.2	64.8	3.7
BoS	✓		81.1	48.2	79.6	41.6	**80.7**	30.1	37.9	4.3	**66.0**	15.5
KVQ-FH	✓		80.8	45.9	79.4	39.1	80.6	26.9	37.5	4.3	**66.0**	15.6
提案手法 (表層)	✓		**82.4**	**64.7**	**80.7**	**56.0**	80.3	**55.6**	34.5	7.4	64.7	25.1
HiCE(文脈)		✓	77.4	38.9	72.9	37.8	75.7	34.0	32.7	1.2	63.1	10.7
提案手法 (文脈)		✓	78.7	47.3	75.4	49.1	76.9	46.4	34.0	1.9	64.0	21.3
提案手法 (表層 + 文脈)	✓	✓	82.2	62.1	80.6	54.0	79.9	44.4	34.7	**13.6**	64.0	**33.6**

	表層	文脈	**BC2**		**BC4**		**BC5**		**NCBI**	
			ALL	OOV	ALL	OOV	ALL	OOV	ALL	OOV
Single-UNK			54.5	29.6	68.6	17.5	77.3	20.8	70.8	27.0
BoS	✓		55.3	29.5	69.2	22.6	77.7	18.7	71.1	30.1
KVQ-FH	✓		55.0	29.7	69.1	22.0	77.6	15.6	71.1	26.6
提案手法 (表層)	✓		60.1	54.9	70.5	38.0	**78.4**	55.8	70.9	41.7
HiCE(文脈)		✓	54.8	**60.6**	66.4	31.7	74.7	46.1	65.1	40.4
提案手法 (文脈)		✓	54.9	60.6	68.0	39.2	75.5	46.3	67.8	44.8
提案手法 (表層 + 文脈)	✓	✓	**60.8**	58.6	**70.8**	**42.0**	**78.4**	**59.8**	**71.9**	**53.4**

表 13　品詞タグ付けと固有表現抽出の結果の比較（単語埋め込み：**fastText**、モデル：w/o LSTM）。

文脈を利用する手法の効果が薄いと考えられる。一方、w/o LSTM においては、固有表現抽出タスクにおいて表層と文脈を組み合わせるモデルの性能が向上した。固有表現抽出タスクが意味的な情報を捉える必要があり、品詞タグ付けは形態素の情報から解くことができるためであると考えられる。最後に、w/o LSTM に対して w/ LSTM の精度が高いことから、タスクの訓練データからタスク特化の単語表現を学習することがより効果的であることが示唆される。

T-POS	~HAPPY/Adjective B-DAY TAYLOR !!! ...					
	近傍					出力
BoS	SHAPE	SHAPED	LOOSE	OVAL	CARAT	Interjection
提案手法	HAPPPY	PMHAPPY	BELATED	AMHAPPY	HAPY	Adjective
Rare-NER	is that Mauro renallo/I-person ?					
	近傍					出力
BoS	renal	end-stage	uremia	glomerular	kidney	O
提案手法	Ranallo	Bednarski	Prazak	DiGiacomo	Mathhew	I-person
BC2GM	... type and variant oCRF1/B-GENE .					
	近傍					出力
BoS	MF1	MF2	R127	Osvald	Parading	O
提案手法	CRF1	NHERF1	LRP1	FHL2	p75NTR	B-GENE
BC5CDR	... pill of amlodipine / benazapril/B-Chemical 10 / 5 mg .					
	近傍					出力
BoS	clinoril	diabeta	dynapen	preventatively	polycillin	O
提案手法	benazepril	Benazepril	enalapril	Lotensin	amlodipine	B-Chemical

表 14　外的評価の出力例（単語埋め込み：**GloVe**）。

6.4　考察

次に、外的評価の考察として、各手法の出力例の比較と綴り誤り訂正モデルを適用した場合について順に説明する。

6.4.1　外的評価の出力例

エラー分析として外的評価の各タスクで単語埋め込みとモデルの出力を分析した。まず、表 14 に単語埋め込みが **GloVe** の時の **BoS** と提案手法の外的評価における出

ARK	What's/nominal+verbal scarier than fake blood , ...					
	近傍					出力
BoS	Whattup	Whattya	Whatthe	Whatya	Whatton	nominal+verbal
提案手法	What	Whatt	.What	Whatya	Whattya	pronoun
BC2GM	... and apocytochrome/B-GENE f proteins has been described ...					
	近傍					出力
BoS	cytochromes	cytochrome	Cytochrome	Cytochromes	flavoproteins	B-GENE
提案手法	autochrome	Autochrome	autochromes	photochrom	orthochromatic	O
BC4CHEMD	A new chromone/B-Chemical from the leaves of ...					
	近傍					出力
BoS	chromonic	pheromone	chlorin	fibromuscular	heteromorphic	B-Chemical
提案手法	chromosones	chromosone	chromosomes	chromsome	chromosomal	O
BC5CDR	··· and followed by apomophine/B-Chemical injection ...					
	近傍					出力
BoS	apomorphine	Apomorphine	levodopa	L-DOPA	apomorphine-induced	B-Chemical
提案手法	apomorphic	apomorphine	apomorphies	apomorphy	apomorphine-induced	O

表 15　外的評価の出力例（単語埋め込み：**fastText**）。

力例の内、提案手法が正解した例と、そのときの未知語の埋め込み表現の近傍の既知語を示す。表中の入力文の下線部は未知語と正解ラベルを示している。その下に、各手法が計算した未知語の埋め込み表現の近傍の既知語と、その未知語について後段のモデルが出力したラベルを示す。まず、1 つ目の T-POS の例では、Happy の表記ゆれである ∼HAPPY に対して、**BoS** が品詞を誤っており、提案手法が正しい品詞を推定できている。**BoS** の近傍にはサブワード HAP を共有する単語が見られることから、このサブワードの影響を受けたと考えられる。次に、2 つ目の Rare=NER の例では、**BoS** が人名である Ranallo を捉えられていない。**BoS** の近傍語には固有名詞とサブワードを共有する普通名詞 renal が見られ、この単語に影響を受けたと考えられる。次に、3 つ目の BC2GM の例では、たんぱく質の ovine CRF1 の略称である oCRF1 を **BoS** が正しく捉えられていない。対して、提案手法では CRF1 を参照することによってより元の意味に近い埋め込みを計算できている。4 つ目の BC5CDR の例では、benazepril の綴り誤りである benazapril に対して、提案手法が正しい綴りの単語を参照している。これらの例から、サブワード埋め込みを用いる手法では、未知語と既知語が共有するサブワードにおいて短く高頻度なサブワードによる影響を受け、長く低頻度なサブ

ワードを考慮することができない傾向にあると考えられる。

次に、表 15 に単語埋め込みが **fastText** の時の **BoS** と提案手法の外的評価における出力例の内、提案手法が誤った例を示す。1 つ目の ARK の例では、nominal+verbal である `What's` に対して、提案手法では `What` を参照しており、後段のモデルにおいて代名詞であるという誤った判断に繋がっている。これは元の埋め込み表現の空間において意味を変化させない短い接尾辞（`s,'s` 等）を重視しないことが影響しており、内的評価においても同様の傾向が見られた。2 つ目の BC2GM の例では、たんぱく質である `apocytochrome` を提案手法が正しく捉えられていない。表層が近く全く異なる意味である `autochrome` を参照しており、このことが出力誤りに繋がったと考えられる。同様に 3 つ目の BC4CHEMD の例では、化合物である `chromone` に対して、提案手法が関連の無い単語である `chromosones` を参照しており、誤った出力になっている。4 つ目の BC5CDR の例では、`apomorphine` の綴り誤りである `apomophine` に対して、提案手法が正しい綴りの `apomorphine` に加えて綴りが近く異なる意味である `apomorphic` を参照しており、出力誤りに繋がっている。これらの例から、提案手法では短い接辞の差異を埋め込み表現に反映することができていないことや、異なる意味の既知語を参照することによる影響を受けやすいと考えられる。特に同じ文脈やトピックを参照する手法 [32,37] を導入することによってこれらの問題に対処できると考えられる。

6.4.2　綴り誤り訂正を適用した場合

綴り誤りに関する手法を比較するために、外的評価に用いたデータセットに綴り誤り訂正を適用した場合について説明する。表 16 に綴り誤り訂正モデルを適用した場合の各タスクの OOV の精度を示す。単語埋め込みは **GloVe** を用い、各タスクの未知語の埋め込み表現のモデルには **Single-UNK** を用いた。綴り誤り訂正にはオープンソースの綴り誤り訂正モデルである After The Deadline (ATD)[*10]を用い、テストデータのテキストに綴り誤り訂正を適用した。表 16 の OOV 割合はテストデータ中の **GloVe** に対する未知語の割合を表し、OOV 精度は綴り誤り訂正を適用する前の未知語に対する精度を表す。外的評価と同様に、品詞タグ付けでは未知語のラベルの精度

[*10] `https://www.afterthedeadline.com/`

		ARK	**T-POS**	**DCU**	**Rare-NER**	Multi-NER
OOV 割合	訂正前	22.2	15.0	9.4	17.9	41.3
	訂正後	21.5	14.7	9.2	15.8	41.3
OOV 精度	訂正前	55.5	56.7	62.9	6.7	29.1
	訂正後	55.7	56.7	61.3	4.1	31.6

		BC2GM	**BC4CHEMD**	**BC5CDR**	**NCBI-Disease**
OOV 割合	訂正前	14.1	18.6	5.4	5.2
	訂正後	7.3	14.0	4.0	3.8
OOV 精度	訂正前	77.6	74.5	64.4	68.5
	訂正後	76.7	71.9	65.8	57.6

表 16　綴り誤り訂正を用いた場合の外的評価。

であり、固有表現抽出では未知語を含むエンティティの F1 値である。

どのデータセットにおいても、綴り誤り訂正によって未知語の割合が減少した。しかし、特に生物医学ドメインにおいて未知語に関する精度が低下した。このことから、綴り誤り訂正によって、一般的な綴り誤りが正しく訂正される一方で、固有名詞が普通名詞などに置き換えられることによって性能が低下したと考えられる。

第 7 章

結論

本研究では、未知語が多く出現するコーパスにおける言語処理技術の高度化を目的とし、単語埋め込みの語彙に対する未知語を適切に処理することに取り組んだ。未知語の埋め込み表現を計算するために、既知語との表層の類似度を学習する手法を提案した。計算した未知語の埋め込み表現を評価するために内的評価と外的評価を行い、既存手法と提案手法の性能を実験的に比較した。実験の結果として、表層の情報を用いる手法では、サブワード埋め込みを学習する手法と比較して同程度かより高精度な埋め込み表現を計算できることを確認した。また、文脈の情報を用いる手法では、表層の類似する単語の文脈を用いて利用する文脈の数を増やすことによる効果が確認された。

今後の課題としてサブワードによる分かち書きを利用するモデルへの適用が挙げられる。BERT を始めとする文脈を考慮した単語表現を事前学習するモデルでは、サブワードによる分かち書きを用いており、綴り誤りに影響を受けやすいことが指摘されている [20]。そのような状況において、表層の類似性を用いることで、綴り誤りにより頑健なモデルが可能になると考えられる。さらに、語彙を固定しない分かち書きモデル [59–61] と組み合わせて、綴り誤り訂正と分かち書きを同時に行う手法等の方向性が考えられる。また、新語の中でも特に頭字語等の表層から意味を類推できない未知語に対して、外部知識を参照する手法 [43–45] を組み合わせて、様々なタイプの未知語に適応できる単一の手法が考えられる。さらに、このように単語埋め込みを on-the-fly で計算するモデルを運用する上ではそのサイズが問題になると考えられるため、埋め込み表現のサイズを削減する手法 [62–64] を効果的に適用することが重要であると考えられる。

謝辞

本研究に取り組むにあたり、本当に多くの方々にご指導ご支援を賜りました。

まず、吉永直樹准教授に感謝申し上げます。本来の指導教員でないにもかかわらず、ご指導して頂き大変感謝しております。お忙しい中、論文執筆やポスター作成、発表資料の修正などご指導していただきました。特に、研究の方針から原稿の言葉使いまで懇切丁寧に指導していただきました。次に、豊田正史教授に感謝申し上げます。ミーティングや個別にご相談した際に、ご助言やご指摘をいただきましてありがとうございます。折りに触れて研究の相談に応じていただき、適切なご助言をしてくださいました。次に、喜連川優教授に感謝申し上げます。素晴らしい研究室環境を用意していただき、不自由なく計算機を用いた実験を行うことができました。

次に、研究室の先輩方に感謝申し上げます。ミーティングなどで的を射たご指摘をいただきました。特に、佐藤翔悦先輩には長時間に亘って研究の内容ついて議論していただき、感謝致します。そして、研究室の同期・後輩方に感謝申し上げます。異なる研究テーマについてお話を伺うことが大変勉強になりました。さらに、個別に質問に答えていたいただいた際には、自分の研究に役立つ知見を得る良い機会になりました。最後に、日々お世話になりました研究室の諸先生方、秘書・経理の方々に感謝を申し上げます。

2020 年 1 月 30 日

参考文献

[1] Jinman Zhao, Sidharth Mudgal, and Yingyu Liang. Generalizing word embeddings using bag of subwords. In *EMNLP*, 2018.

[2] Shota Sasaki, Jun Suzuki, and Kentaro Inui. Subword-based compact reconstruction of word embeddings. In *NAACL-HLT*, 2019.

[3] Mohammad Taher Pilehvar, Dimitri Kartsaklis, Victor Prokhorov, and Nigel Collier. Card-660: Cambridge rare word dataset - a reliable benchmark for infrequent word representation models. *EMNLP*, 2018.

[4] Michael Flor, Michael Fried, and Alla Rozovskaya. A benchmark corpus of english misspellings and a minimally-supervised model for spelling correction. In *BEA@ACL*, 2019.

[5] Ziniu Hu, Ting Chen, Kai-Wei Chang, and Yizhou Sun. Few-shot representation learning for out-of-vocabulary words. In *ACL*, 2019.

[6] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, 2013.

[7] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *EMNLP*, 2014.

[8] Zellig S. Harris. Distributional structure. 1981.

[9] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. In *TACL*, 2016.

[10] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones,

Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017.

[11] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. In *ICLR*, 2017.

[12] Yi Zhu, Ivan Vulic, and Anna Korhonen. A systematic study of leveraging subword information for learning word representations. In *NAACL-HLT*, 2019.

[13] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *ACL*, 2015.

[14] Taku Kudo. Subword regularization: Improving neural network translation models with multiple subword candidates. In *ACL*, 2018.

[15] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *NAACL-HLT*, 2018.

[16] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, Vol. 9, pp. 1735–1780, 1997.

[17] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 2019.

[18] Naoaki Okazaki and Jun'ichi Tsujii. Simple and efficient algorithm for approximate dictionary matching. In *COLING*, 2010.

[19] Zhenglu Yang, Jianjun Yu, and Masaru Kitsuregawa. Fast algorithms for top-k approximate string matching. In *AAAI*, 2010.

[20] Danish Pruthi, Bhuwan Dhingra, and Zachary Chase Lipton. Combating ad-

versarial misspellings with robust word recognition. In *ACL*, 2019.

[21] Ryan Cotterell, Hinrich Schütze, and Jason Eisner. Morphological smoothing and extrapolation of word embeddings. In *ACL*, 2016.

[22] Geewook Kim, Kazuki Fukui, and Hidetoshi Shimodaira. Segmentation-free compositional n-gram embedding. In *NAACL-HLT*, 2018.

[23] Yang Xu, Jiawei Liu, Wei Yang, and Liusheng Huang. Incorporating latent meanings of morphological compositions to enhance word embeddings. In *ACL*, 2018.

[24] Benjamin Heinzerling and Michael Strube. Sequence tagging with contextual and non-contextual subword representations: A multilingual evaluation. In *ACL*, 2019.

[25] Yang Xu, Jiasheng Zhang, and David Reitter. Treat the word as a whole or look inside? subword embeddings model language change and typology. 2019.

[26] Yuval Pinter, Robert Guthrie, and Jacob Eisenstein. Mimicking word embeddings using subword rnns. In *EMNLP*, 2017.

[27] Bora Edizel, Aleksandra Piktus, Piotr Bojanowski, Rui Ferreira, Edouard Grave, and Fabrizio Silvestri. Misspelling oblivious word embeddings. In *NAACL-HLT*, 2019.

[28] Tomoya Mizumoto and Ryo Nagata. Analyzing the impact of spelling errors on pos-tagging and chunking in learner english. In *NLP-TEA@IJCNLP*, 2017.

[29] Aurélie Herbelot and Marco Baroni. High-risk learning: acquiring new word vectors from tiny data. In *EMNLP*, 2017.

[30] Andrew Kyle Lampinen and James L. McClelland. One-shot and few-shot learning of word embeddings. *ArXiv*, Vol. abs/1710.10280, , 2018.

[31] Tianfan Fu, Cheng Zhang, and Stephan Mandt. Continuous word embedding fusion via spectral decomposition. In *CoNLL*, 2018.

[32] Alexandre Kabbach, Kristina Gulordava, and Aurélie Herbelot. Towards incremental learning of word embeddings using context informativeness. In *ACL*, 2019.

[33] Qianchu Liu, Diana McCarthy, and Anna Korhonen. Second-order contexts from lexical substitutes for few-shot learning of word representations. In **SEM@NAACL-HLT*, 2019.

[34] Timo Schick and Hinrich Schütze. Learning semantic representations for novel words: Leveraging both form and context. In *AAAI*, 2019.

[35] Angeliki Lazaridou, Marco Marelli, and Marco Baroni. Multimodal word meaning induction from minimal exposure to natural text. *Cognitive science*, Vol. 41 Suppl 4, pp. 677–705, 2017.

[36] Mikhail Khodak, Nikunj Saunshi, Yingyu Liang, Tengyu Ma, Brandon Stewart, and Sanjeev Arora. A la carte embedding: Cheap but effective induction of semantic feature vectors. In *ACL*, 2018.

[37] Timo Schick and Hinrich Schütze. Attentive mimicking: Better word embeddings by attending to informative contexts. In *NAACL-HLT*, 2019.

[38] Minlong Peng, Qi Zhang, Xiaoyu Xing, Tao Gui, Jinlan Fu, and Xuanjing Huang. Learning task-specific representation for novel words in sequence labeling. In *IJCAI*, 2019.

[39] Shaonan Wang, Jiajun Zhang, and Chengqing Zong. Associative multichannel autoencoder for multimodal word representation. In *EMNLP*, 2018.

[40] Ignacio Iacobacci and Roberto Navigli. Lstmembed: Learning word and sense representations from a large semantically annotated corpus with long short-

term memories. In *ACL*, 2019.

[41] Koki Washio, Satoshi Sekine, and Tsuneaki Kato. Bridging the defined and the defining: Exploiting implicit lexical semantic relations in definition modeling. In *EMNLP/IJCNLP*, 2019.

[42] Shonosuke Ishiwatari, Hiroaki Hayashi, Naoki Yoshinaga, Graham Neubig, Shoetsu Sato, Masashi Toyoda, and Masaru Kitsuregawa. Learning to describe unknown phrases with local and global contexts. In *NAACL-HLT*, 2019.

[43] Dzmitry Bahdanau, Tom Bosc, Stanislaw Jastrzebski, Edward Grefenstette, Pascal Vincent, and Yoshua Bengio. Learning to compute word embeddings on the fly. In *CoRR*, 2018.

[44] Nigel Collier and Mohammad Taher Pilehvar. Inducing embeddings for rare and unseen words by leveraging lexical resources. In *EACL*, 2017.

[45] Ziyi Yang, Chenguang Zhu, Vin Sachidananda, and Eric F Darve. Embedding imputation with grounded language information. In *ACL*, 2019.

[46] Thang Luong, Richard Socher, and Christopher D. Manning. Better word representations with recursive neural networks for morphology. In *CoNLL*, 2013.

[47] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ICLR*, 2014.

[48] Kevin Gimpel, Nathan Schneider, Brendan T. O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. Part-of-speech tagging for twitter: Annotation, features, and experiments. In *ACL*, 2011.

[49] Alan Ritter, Sam Clark, Mausam, and Oren Etzioni. Named entity recognition in tweets: An experimental study. In *EMNLP*, 2011.

[50] Jennifer Foster, ÖzlemÇetinoglu, Joachim Wagner, Joseph Le Roux, Stephen Hogan, Joakim Nivre, Deirdre Hogan, Josef van Genabith. #hardtoparse: Pos tagging and parsing the twitterverse. In *Analyzing Microtext*, 2011.

[51] Derczynski Leon, Eric Nichols, Marieke van Erp, and Nut Limsopatham. Results of the wnut2017 shared task on novel and emerging entity recognition. In *NUT@EMNLP*, 2017.

[52] Qi Zhang, Jinlan Fu, Xiaoyu Liu, and Xuanjing Huang. Adaptive co-attention network for named entity recognition in tweets. In *AAAI*, 2018.

[53] Larry Smith, Lorraine K. Tanabe, Rie Johnson nee Ando, Cheng-Ju Kuo, I-Fang Chung, Chun-Nan Hsu, Yu-Shi Lin, Roman Klinger, Christoph M. Friedrich, Kuzman Ganchev, Manabu Torii, Hongfang Liu, Barry Haddow, Craig A. Struble, Richard J. Povinelli, Andreas Vlachos, William A. Baumgartner, Lawrence E. Hunter, Bob Carpenter, Richard Tzong-Han Tsai, Hong-Jie Dai, Feng Liu, Yifei Chen, Chengjie Sun, Sophia Katrenko, Pieter Adriaans, Christian Blaschke, Rafael Torres, Mariana Neves, Preslav Nakov, Anna Divoli, Manuel J. Maña-López, Jacinto Mata, and W. John Wilbur. Overview of biocreative ii gene mention recognition. *Genome Biology*, Vol. 9, pp. S2 – S2, 2008.

[54] Martin Krallinger, Florian Leitner, Obdulia Rabal, Miguel Vazquez, Julen Oyarzábal, and Alfonso Valencia. Chemdner: The drugs and chemical names extraction challenge. In *Journal of Cheminformatics*, 2015.

[55] Chih-Hsuan Wei, Yifan Peng, Robert Leaman, Allan Peter Davis, Carolyn J. Mattingly, Jiao Li, Thomas C. Wiegers, and Zhiyong Lu. Assessing the state of the art in biomedical relation extraction: overview of the biocreative v chemical-disease relation (cdr) task. *Database : the journal of biological databases and curation*, Vol. 2016, , 2016.

[56] Rezarta Islamaj Dogan, Robert Leaman, and Zhiyong Lu. Ncbi disease corpus: A resource for disease name recognition and concept normalization. *Journal of biomedical informatics*, Vol. 47, pp. 1–10, 2014.

[57] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. In *HLT-NAACL*, 2016.

[58] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *ICLR*, 2016.

[59] Edouard Grave, Armand Joulin, and Nicolas Usunier. Improving neural language models with a continuous cache. In *ICLR*, 2016.

[60] Kazuya Kawakami, Chris Dyer, and Phil Blunsom. Learning to create and reuse words in open-vocabulary neural language modeling. In *ACL*, 2017.

[61] Tatsuya Hiraoka, Hiroyuki Shindo, and Yuji Matsumoto. Stochastic tokenization with a language model for neural text classification. In *ACL*, 2019.

[62] Patrick H. Chen, Si Si, Yang D. Li, Ciprian Chelba, and Cho-Jui Hsieh. Groupreduce: Block-wise low-rank approximation for neural language model shrinking. *NeurIPS*, 2018.

[63] Yukun Ma, Patrick H. Chen, and Cho-Jui Hsieh. Mulcode: A multiplicative multi-way model for compressing neural language model. In *EMNLP/IJCNLP*, 2019.

[64] Dinghan Shen, Pengyu Cheng, Dhanasekar Sundararaman, Xinyuan Zhang, Qian Yang, Meng Tang, AsliÇelikyilmaz, Lawrence Carin. Learning compressed sentence representations for on-device text processing. In *ACL*, 2019.

発表文献

査読無し国内会議

1. 福田 展和、佐藤 翔悦、吉永 直樹、喜連川 優、
   "Wikipedia の内部リンクを用いた弱教師あり共参照解析"、
   言語処理学会第 25 回年次大会 (NLP2019)、名古屋、2019。

査読無し国内研究会

1. 福田 展和、吉永 直樹、喜連川 優、
   "複数のサブワード分割と表層類似語を用いた未知語の単語分散表現の生成"、
   NLP 若手の会第 14 回シンポジウム (YANS2019)、札幌、2019。

2. 福田 展和、吉永 直樹、喜連川 優、
   "複数のサブワード分割と表層類似語を用いた未知語の単語分散表現の生成"、
   東京大学音声・言語・コミュニケーション研究会、東京、2019。

査読無し国内会議（予定）

1. 福田 展和、吉永 直樹、喜連川 優、
   "既知語との表層類似性に基づく未知語の埋め込み表現の計算"、
   言語処理学会第 26 回年次大会 (NLP2020)、水戸、2020。