Vol.2018-NL-237 No.8
2018/9/26

構文情報を陽に与えたときの LSTM による内部表現について

岡本 千尋[1,a)]　内海 慶[2,b)]　持橋 大地[3,c)]

概要：長短期記憶リカレントニューラルネット (LSTM) は，構文情報に代表されるような文中の有用な長期依存関係を捉えることにより，高精度な言語モデルを学習することができることが知られている．しかし逆に，一般的にどの程度構文情報を学習できているのかや，どのように構文情報がベクトルとして RNN 内に表現されるかについては，まだ十分に研究されていないのが現状である．そこで，我々はその初歩として，英語における句構造をあらわす構文木を線形化し文として与えて学習させたときに，RNN 内で構文情報がどのようにエンコードされるかについて詳細に分析した．その結果，L_1 正則化を用いることで，例えば RNN 内の内部ベクトルのうち少数の要素の値が VP, NP など句構造の各タグごとのネストの深さと非常に高い相関を持つこと，および，同じく少数の要素の値から，VP，NP などの句の内外にいることを高精度で線形分離できること，などがわかった．

Toward Understanding of How LSTMs Internally Represent Syntactic Information

Chihiro Okamoto[1,a)]　Kei Uchiumi[2,b)]　Daichi Mochihashi[3,c)]

Abstract: Long Short-Term Memory recurrent neural networks (LSTMs) are known to be able to capture informative long-term dependencies in a sentence such as syntactic information, and to learn highly accurate language models. However, it is not yet sufficiently studied to what extent syntax information can be learned and how it is represented in the hidden vectors of RNNs. We analyze how it is represented in the context vectors of LSTMs from sentences in which syntactic information is explicitly given. The syntax trees representing phrase structures in English are linearized and fed as sentences to LSTMs for learning. We show empirically that for some element in the context vector, the activation value is highly correlated with the nest depth of the tagged phrase structures such as VP and NP. It also shown that the linear sums of a few elements in the context vector are extremely highly correlated with the nest depths of the tagged phrase structures. In addition, using logistic regression with L_1 regularization, it is able to predict with extremely high accuracy whether a word in a sentence is inside or outside a phrase such as VP or NP, from a small number of elements in the context vector.

1. はじめに

長短期記憶リカレントニューラルネット (LSTM) [5] は，自然言語などの記号化された時系列データに対して，勾配消失をうまくコントロールし，長期の依存関係を取捨選択することによって，高精度な言語モデルを学習することができることが知られている．その一方で，どのような形で長期依存関係を捉えているかについては，いまだ完全には解明されていない．例えば，U. Khandewal ら [6] は，Penn Treebank [12] や Wiki などのデータセットを用い，LSTM を用いた言語モデルを構築し，過去の単語と時系列の意味での距離が，次単語予測にどのような影響を与えるかを評価している．また，G.Weiss ら [14] は，LSTM や GRU [4] など RNN を用いた言語モデルを学習させ，それをオラク

[1] 東京工科大学大学院 バイオ・情報メディア研究科
[2] デンソーアイティーラボラトリ
[3] 統計数理研究所 数理・推論研究系
[a)] shibatachh@stf.teu.ac.jp
[b)] kuchiumi@d-itlab.co.jp
[c)] daichi@ism.ac.jp

ルとして使うことで，質問学習の枠組みによりオートマトンを獲得する手法を提案している．しかし，長期依存の代表的な例として，自然言語における句構造などの構文情報が挙げられるが，より一般的に，構文情報による長期依存が，LSTM を用いた言語モデルにおいて，どの程度学習できるかについて，まだ十分に研究されているとは言えない．さらに，どのように構文情報がベクトルとして RNN 内に表現されるかについても，研究がされていないのが現状である．そこで我々は，その初歩として，英語における句構造をあらわす構文木を線形化し陽に与えて学習させたときに，RNN 内で構文情報がどのようにエンコードされるかを調べる．とくに，句構造の種類ごとのネストの深さに対して，LSTM の文脈ベクトルの要素の値 (またはその線形和) が線形に変化すると仮定して，分析を行う．

2. LSTM を用いた言語モデル

本研究では一層の LSTM による言語モデルを考える．h_t を LSTM の時刻 t での**入出力ベクトル**，c_t を**文脈ベクトル**，$e(w_t)$ を単語 w_t の埋め込みベクトルとする．$lstm(R, h, c, e)$ を，パラメータ R, 入力として h,c,e を与えたときの LSTM の出力関数の出力とする．すなわち，

$$(h_{t+1}, c_{t+1}) = lstm(R, h_t, c_t, e(w_t)).$$

言語モデルは，t 番目までの単語列が与えられたときの次の単語の出現確率で表され，上記を用いて，

$$p(w_{t+1}|w_1, \cdots, w_t) = p(w_{t+1}|c_t, h_t, w_t) = s(Wh_{t+1} + b)$$

となる．ここで，$s()$ はソフトマックス (多値ロジスティック) 関数, W, b はそのパラメータである．上式のように，$t-1$ までの単語列は確率の条件部分に陽に現れず，ベクトル c_t および h_t に何らかの形でその情報が表現されているといえる．c_t と h_t のどちらに過去の単語の情報が表現されているかについては，次のようなことが知られている．まず，定性的に言って，遠い過去の単語，つまり単語 w_{t-k} のうち k が大きいものは，k の値に比例した多数回の活性化関数への代入により勾配消失を起こすため，h_t にはほとんど影響しないと考えられている．代わりに，それらの情報は c_t に含まれており，c_t を介して長期の依存関係が捉えられていると考えられている．c_t が k の大きい w_{t-k} の影響を受ける理由としては，一般に c_t と c_{t+1} の間には，忘却ゲートとのアダマール積 (要素ごとの積) および更新ベクトルの加算しかなく，忘却ゲートの値が 1 に近い限り, k の値が大きくなっても勾配消失を起こさないためとされる．

3. 学習用データ

学習，テスト用のデータとしては，Penn Treebank [9][12] の WSJ コーパス全体より，構文木つきのテキスト約 49,000 文を用いる．この構文木は文の句構造を表しており，すべてのノードにタグが付いており，リーフについているタグは単語の品詞となっている．例えば，“a nonexecutive director” という部分列に対する部分木のノードには NP のタグが付き, その 3 つの子ノード（リーフ）にはそれぞれ DT, JJ, NN のタグが付いている．今回は，構文木により表現される構文を陽に与えて学習させたとき，LSTM がどのようにそれを内部で表現するようになるかを調べることが目的である．したがって，まず，単語を含まない構文構造のみを取り出し，次の 2 通りの形へ線形化を行った．

(1) ノードについているタグを無視し，‘(’ および ‘)’ の 2 単語のみから構成される文,

(2) ノードについているタグを残し, タグ付きの ‘(タグ’ および ‘タグ)’ から構成される文.

例えば，前述の “a nonexecutive director” に対する構文木に対しては，前者は “(( ) ( ) ( ))” となり，後者は “(NP (DT DT) (JJ JJ) (NN NN) NP)” となる．後者は，わかりにくいが，‘(NP’ や ‘JJ)’ などが 1 つの単語である．なお，すべての文の前後に，文の開始と終了を表す単語 (以降 ‘–ST–’,‘–ED–’) を挿入する．文の開始や終了を陽に表現することによって，それらを LSTM がどう表現するかや，正しく文の終了を判定するかを調べることができるためである．(1) のモデルでは単語の総数は 4 であり, (2) のモデルでは 142 となった．

上記 (1)(2) では，単語のかわりにその品詞のみが情報として残っている．そこで，別のデータとして，句構造のうち NN や JJ など品詞を表すタグが付いているものの代わりに単語を残したデータについても実験を行った．

(3) タグのない ‘(’ および ‘)’ および単語から構成される文,

(4) タグ付きの ‘(’ および ‘)’ および単語から構成される文 (ただし品詞のタグは除く).

例えば，“Pierre Vinken will join the board .” であれば，(3),(4) はそれぞれ，“((Pierre Vinken) (will (join (the board))) .)” および，“(S (NP Pierre Vinken NP) (VP will (VP join (NP the board NP) VP) VP) . S)” のようになる．

前処理としては，全て小文字にしたあと，数字なども含め単語の種類数が 10,000 未満となるよう，6 回以下しか出現しない単語はすべて品詞に置き換えた．その結果，文中の単語の出現数全体の 1.8%ほどが品詞に置き換えられ，品詞も含めた単語の種類数は 10,011 となった．

4. 学習とテスト

前述の LSTM を用いた言語モデルをそれぞれのデータで学習させる．詳細は次のとおりである．まず，長さ 200 以上の文は削除する．その結果，文の数は 49,208 文から 48,609 文と 1%ほど減少した．残った文のうち 1/10 をテ

ストデータ，残りを訓練データとする[*1]．単語の埋め込み次元および LSTM の c, h の次元はすべて同じとし，(1) のデータに対しては 100, (2) に対しては 200, (3),(4) に対しては単語種が多いため 1,000 とする．学習方法としては，誤差関数はクロスエントロピーとし，文ごとにまとめて誤差の総和を計算したのち誤差逆伝搬を行う．また，バッチサイズは 128 とするが，文長をそろえるためのパディングは行わない[*2]．ドロップアウトは c での構文情報の表現を見ることが目的であるため行わない．忘却ゲートの初期値のバイアスは 0 とするなど，パラメータの初期値はすべて用いたライブラリのデフォルトのものを用いた．学習アルゴリズムとしては Adam を用い，学習の繰り返し回数は 50 エポックとした．ただし，(3),(4) については，過学習を防ぐため，テストデータに対する正解率が最大となる時点のものを用いた．

5. 分析の方法

分析の方法としては，まず，文の終わりを表す ‘–ED–’ を含めた，言語モデルとしての単語予測の精度を単語ごとに示す．特に，‘–ED–’ をどの程度予測できているかにより，‘(’ と ‘)’ の数のバランスという，長期的な情報が保持されているかがわかる．

また，LSTM の文脈ベクトルに焦点を当て，タグごとにそれぞれ，次の関係を調べる．

(i) ネストの深さと，文脈ベクトルの各要素の値の関係
(ii) ネストの深さと，文脈ベクトルの複数の要素の線形和との関係
(iii) 句の内外のどちらにあるかの 2 値と，文脈ベクトルの複数の要素の線形和との関係

分析のために，統計的学習手法として (i)(ii) に対しては線形回帰，(iii) に対してはロジスティック回帰を用いる．本研究の目的は，陽に与えた句の情報がどのように文脈ベクトル中に表現されているかを明らかにすることであるので，最も解釈のしやすい線形の手法を用いる．また，(ii) については L_1 正則化を用いて，複数の要素のうち，どの要素が主に影響しているかを調べる．

6. 実験結果と評価

6.1 データ (1) に対する学習結果と分析

6.1.1 予測精度

まず，(1) の括弧データに対する言語モデルの学習結果を述べる．テストデータに対して，次の単語の予測をすべての接頭辞に対して行った．このときの予測結果と正解の混同行列を図 1 に示す．この混同行列は，縦方向，つまり

[*1] テストデータは，WSJ の評価用セットは用いず訓練用セットの中からランダムに選んだ
[*2] 実装では深層学習用のライブラリである Chainer の NStepLSTM クラスを用いた．

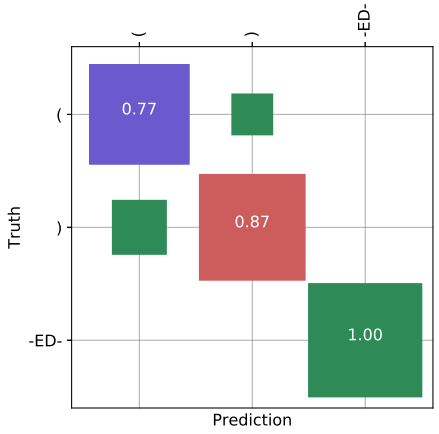

図 1 (1) のテストデータに対する言語モデルの予測結果. タイル中の数字は，予測精度 (precision) を表している．

予測した単語ごとに正規化を行っている．図 1 からまず明らかに言えることは，文の最後を示す’–ED–’ については，recall, precision 両方の意味で例外なく正しく予測できているということである．(1) のデータにおいては，すべての文は文の先頭を除いて，‘(’ の数と ‘)’ の数が等しいとき，またそのときに限り終わる．したがって上記の事実は，その数が等しいかどうかが完全に正しく判定されていることを示している．

‘(’ と ‘)’ については，どちら方向の間違いについても，ともに同数程度，誤分類している事がわかる．予測精度としては高いとは言えないが，句構造のタグや単語を全て除いたため，そもそも真の言語モデルにおいても確率的な振る舞いをするものであるためであると考えられる．

6.1.2 文脈ベクトルの分析

LSTM の学習では，過去の履歴 $(w_0, \cdots, w_{t-1})$ が文脈ベクトル c_t に何らかの形で表現されているはずであるが，どのようにエンコードされているかについて，ここでは追求する．前節で述べたように，(1) のデータに対する学習では，文の終わりが正確に予測されていた．‘(’ の数と ‘)’ の数の差を常に記憶していて，それが何らかの形で c_t 内に埋め込まれていると考えるのが自然である．そこで，まず，文脈ベクトル c_t の各要素の値，言い換えると，時刻 t におけるアクティベーションの値と深さの相関係数を調べる．

図 2 上は，要素数，つまり LSTM のベクトルのサイズは 100 であるので，その数の相関係数が得られるが，それらをすべてヒストグラムとしてプロットしたものである．相関係数の最大値は 0.998 と非常に高いものであり，また，図の示すとおり，相関係数が 1 に極めて近いものが，いくつか存在することがわかる．図 2 下は，相関係数が最大の要素を取り，構文木の深さとアクティベーションの値との関係をプロットしたものである．ほぼ直線上に乗っていることがわかる．深さ 0 においてのみ，アクティベーションの値が 約 -1 のときと約 -2 のときに分かれている事がわ

かる．これは，後述のように，−1 付近のアクティベーションは 文の開始を表す単語である ’–ST–’ を入力したとき，つまり $t = 0$ のときの値であると考えられる．それに対して，−2 付近のものは，文の終わり，つまり最終的に ‘(’ の数と ‘)’ の数が等しくなったときの値であると考えられる．

図 3 は，テストデータの中から長さ 100 未満の数十文をとって，相関係数が最大の要素のアクティベーションの値を時系列としてプロットしたものである．数十本のアクティベーションのグラフを重ねて表示している．図より，まず，若干のブレはあるものの，連続値であるこれは，c ベクトルに加算される更新ベクトルの値が tanh 関数や シグモイド 関数の効果で $[-1, 1]$ の範囲となるが，学習によりおおよそ −1,+1 のほぼどちらかを出力するように偏ったことが理由だと考えられる．また，すべてのグラフの開始点 ($t = 0$) におけるアクティベーションの値は当然同一で，−1 付近である．一方，すべてのグラフの右端点は，どのような長さに対しても，アクティベーションの値が −2 付近となっていることがわかる．実際には c ベクトルではなく h ベクトルの値に更に全結合を経て次単語 w_{t+1} の確率が決定されるが，文の終わりについては，文脈ベクトルの相関係数が最大の要素のアクティベーションの値が −2 付近かどうかだけで，殆どの場合判断が可能であると考えられる．

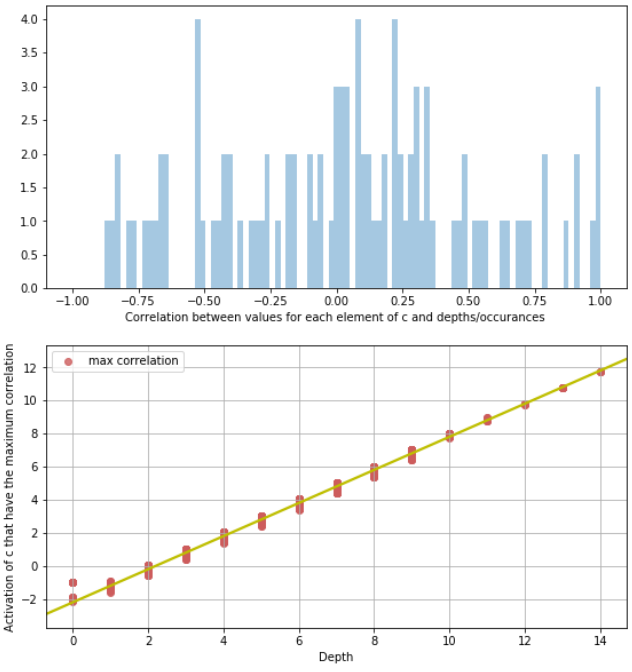

図 2 上: c の各要素の相関係数のヒストグラム．下: 相関係数が最も大きい c の要素のアクティベーションの値と構文木の深さとの関係

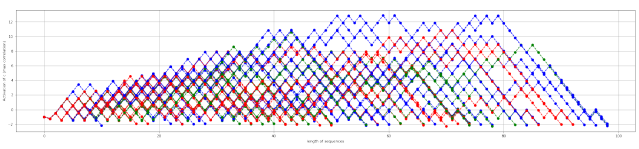

図 3 相関係数が最も大きい c の要素の時系列としての推移．数十ある各グラフはテストデータ中の各文を示している．グラフの着色は各グラフを区別しやすくするため．

6.2 データ (2) に対する学習結果と分析

6.2.1 予測精度

(2) のデータに対する学習結果を述べる．図 4 は，テストデータに対する LSTM を用いた言語モデルの次単語予測の予測結果を表す混同行列である．WSJ 構文データに出現している品詞+ 構文タグの種類数は 70 程度ある．そのため，(2) のデータの単語数は 142 あり，すべてを表示することはできないため，頻度の高い 11 のタグ (単語数では 22 個) とその他，および ‘–ED–’ の合計 24 個の単語についての混同行列となっている．

まず，実験結果 1 の場合と同様に，‘–ED–’ に対する予測に注目すると，recall, precision ともにほぼ 100% 正解していることがわかる．これは，‘–ED–’, ‘–ST–’ 以外の単語を，タグなしの ‘(’ と ‘)’ の 2 種類に分類したとき，その数が等しいかどうかをほぼ完全に正しく判定できていることを示している．単語は各文字を情報として与えられることなどはないため，単語がタグを無視したとき ‘(’ と ‘)’ のどちらに属するかは，単語の ID からはわからないことから，各タグ A に対して，‘(A’ と ‘A)’ の対応関係が自然に学習されたと考えられる．

また，タグ付きの’(’ と ’)’ (図中青および青) の正解率の

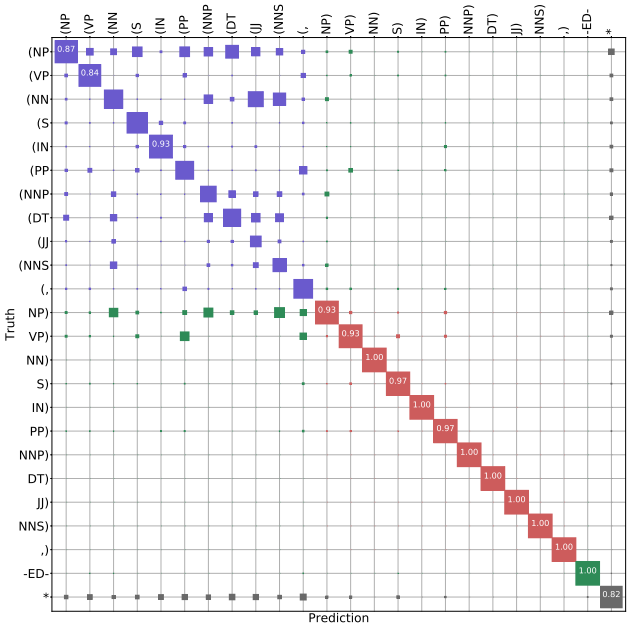

図 4 (2) のテストデータに対する言語モデルの予測結果．図中，単語は頻度の高い単語のみを用い，頻度の低い単語はまとめて ‘*’ で表している． また，タイルの色は，単語の種類を表す．青: ‘(’+タグ，赤: タグ+‘)’，灰: ‘*’，緑: ‘–ED–’ および通常の単語．

違いに注目すると、タグ付きの')' については全てにおいて正解率が 0.9 以上であり，予測精度が'(' よりも高いことがわかる．これは，'(A' と 'A)' の数の対応関係がうまく学習できているというだけでなく，その出現順序についても高い精度で学習できているということを意味する．また，部分木の終了をうまく予測できているということであり，例えば 'NP' など構文木における子ノードの数が変動する場合も，高い精度で予測できていることがわかる．

6.2.2 文脈ベクトルの分析

図 5 は動詞句のタグが付いているような単語，すなわち '(VP' と 'VP)' によるネストの深さと c ベクトルのアクティベーションの関係を示している．図 5 左上は，各要素ごとのアクティベーションの値と深さから得られた合計 200 個の相関係数を，ヒストグラムとして表示したものである．(1) のデータの学習結果とは異なり，相関係数が 1 に極めて近いようなものは存在しないものの，ある程度高い相関を示す要素は存在する．相関係数の最大値は 0.86 であった．相関係数が最大の要素を取り出し，そのアクティベーションと，VP のネストの深さの関係を図 5 左下に示す．深さに対して，ほぼ線形にアクティベーションの値が増えていることが見て取れるものの，深さごとの分散が大きく，この要素のアクティベーションの値が，純粋に VP のネストの深さのみで決定されているとは考えがたい．図 5 右上は，構文木において VP の中に入っているかどうかの 2 値でアクティベーションの値を分類したとき，その 2 つのヒストグラムの重なり部分の割合 (以降，ヒストグラム重複率とよぶ) を各要素ごとに求め，それをヒストグラムとしたものである．ヒストグラム重複率が 1 に近いほど，その要素のみからは，VP の中に入っているかどうかを判別できないことを意味しており，0 に近いほど，判別精度が高くできることを意味している．ヒストグラム重複率の最小値は 0.28 であった．その最小値を与える要素におけるアクティベーションの値を，VP の中に入っているかどうかで分けた二つのヒストグラムを示したものが図 5 右下である．必ずしも図 5 左下に示されている要素と同じ要素が選択されるとは限らない．図 5 右下からもわかるように，分布の重複面積を最小にする要素を選んでも，完全に VP の内外を判別することはできそうにない．

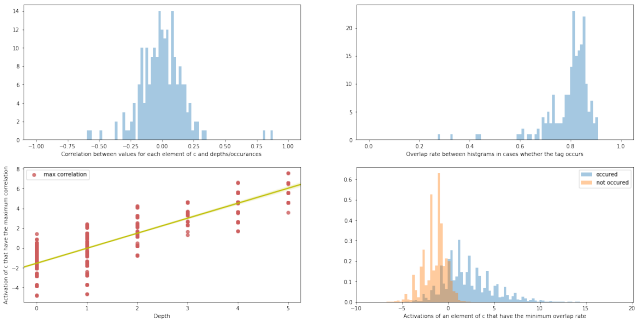

図 5 動詞句 (VP) のネストの深さと各要素のアクティベーション．

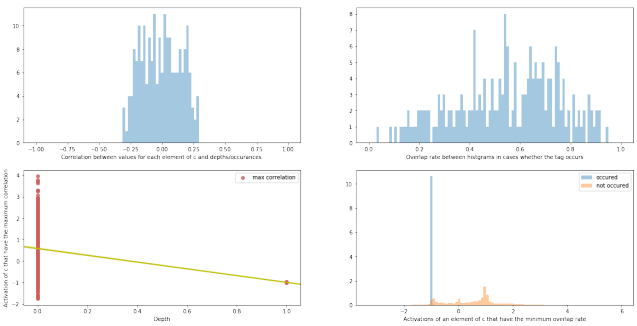

図 6 一般名詞 (NN)(品詞タグ) のみに限定した場合の各要素のアクティベーション．

図 6 は一般名詞のタグが付いているような単語，すなわち '(NN' と 'NN)' によるネストの深さと c ベクトルのアクティベーションの関係を示している．NN は品詞であるため，ネストはせず，更に '(NN' が出現したあとすぐに 'NN)' が出現する．したがって，バイグラムで十分に表現でき，長期の情報の保持は必要ないが，実際には図 6 左上に示すように，c に相関係数の十分に高い要素は存在しないものの，ある要素において，'(NN' を入力した直後には必ず -1 付近に集中している．図 6 右上下にはヒストグラムの重複率の分布と，それが最小となる要素におけるアクティベーションの値の分布を示す．ヒストグラムの重複率は 0.03 であり十分に低いことからも，単語 '(NN' の出現は，c の特定の要素の値を -1 にリセットするような影響を与えていると考えられる．

また，線形回帰の結果の深さの予想値が 0.5 以上かどうかで, VP の内外 (つまり深さが 0 か 1 以上) の判定を行ったところ，正解率は，正則化なしの場合 0.993, L_1 正則化ありの場合は 0.963 であった．ただ，線形回帰は本来分類の方法ではない．したがって，L_1 正則化付きのロジスティック回帰で VP の内外を判定した結果を表 1 に示す．C はペ

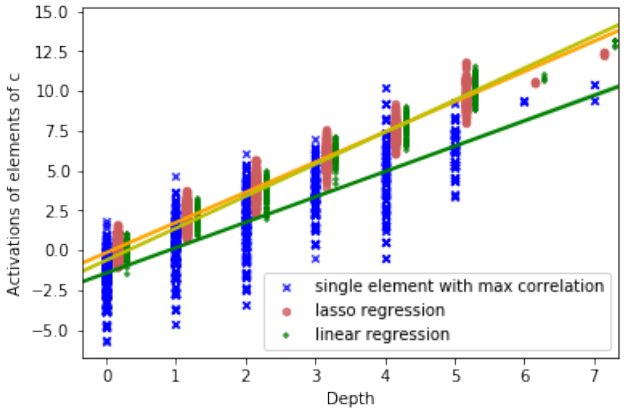

図 7 VP のネストの深さに対する関数としての Lasso 回帰, 線形回帰, および相関係数が最大となる要素のアクティベーションの比較．回帰の結果については，見やすさのために，スケールを揃えて深さのプロット位置を右に少しスライドさせている．

表 1　ロジスティック回帰による VP の内外の判定 (データセット (2)).

C	正解率	非ゼロ要素数 (200 次元中)	非ゼロの割合
0.003	0.996	82	41%
0.001	0.994	56	28%
0.0003	0.991	34	17%
0.0001	0.98	21	11%
0.00003	0.96	8	4%
0.00001	0.91	5	2.5%

ナルティ係数の逆数であり，小さいほど正則化が強くかかることを意味する．

次に，VP において，各要素ごとではなく，c から句構造の深さへの線形回帰によって，どの程度相関が増えるかを調べる．また，相関係数をあまり下げない範囲で，極力係数が非ゼロの要素数を減らしたい．そこで，L_1 正則化を用いた線形回帰である Lasso 回帰を用い，ペナルティ係数を 0.1 とした．その結果を図 7 に示す．要素ごとの相関係数の最大値は 0.865, Lasso 回帰の相関係数は 0.977, 正則化なしの線形回帰の相関係数は 0.990 であり，正則化なしの線形回帰の相関係数が最も高い．一方，係数が非ゼロの要素数は，Lasso 回帰では，当然ペナルティ係数に依存するが，13 であり，正則化なしの回帰の場合 (200) よりも十分に少ない．

6.3　データ (3) に対する学習結果と分析

6.3.1　予測精度

(3) のデータに対する学習結果として，まず図 8 に予測の混同行列を示す．‘–ED–’ がほぼ 100%正解している点は同じである．一方で，データ (1) に対する結果 (図 1) と比べて，‘(’ (青) と ‘)’ (赤) の予測精度が共に大きく向上していることがわかる．これは，単語の存在が手がかりとなって，それらの予測が行われるためと考えられる．単語の予測については，ケースにより大きく異なることがわかる．例えば，‘of’ や ‘to’ などは比較的正確に予測されているものの，‘a’ や　‘in’，‘and’ などは，本来 ‘(’ や ‘)’ であるところを誤って予測しており, 結果として予測の precision が低くなっている．

6.3.2　文脈ベクトルの分析

一方, ‘(’ と ‘)’ のネストの深さをアクティベーションの値として表現するような，文脈ベクトルの要素が存在するかについては，(1) のデータセットに対する実験結果と同様の結果が得られた (図 9)．すなわち，ある要素があって，相関係数 0.9986 という非常に大きい相関を持ち，図 9 下のように，深さとその要素のアクティベーションはほぼ直線上に乗っていることがわかった．

6.4　(4) に対する学習結果

(4) のテストデータに対する予測結果を図 10 に示す．(2)

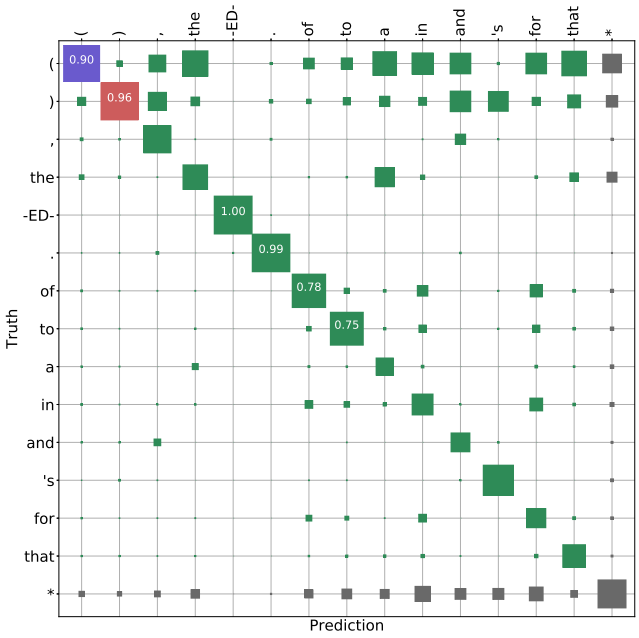

図 8　(3) のテストデータに対する言語モデルの予測結果.

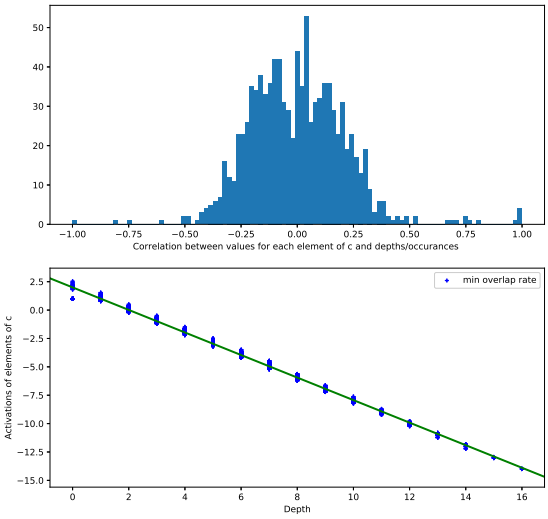

図 9　上: c の各要素の相関係数のヒストグラム．下: 相関係数が最も大きい c の要素のアクティベーションの値と構文木の深さとの関係

の結果と同様，‘–ED–’ に対する予測は 99%以上正解していることがわかる．また，全体的に，‘NP)’ や ‘VP)’ など，部分木の終了をあらわす単語の予測精度が高いことがわかる．

6.5　文脈ベクトルの分析 3

図 11 に, データセット (2) に対する分析と同様に，VP におけるネストの深さと各要素の各アクティベーションの

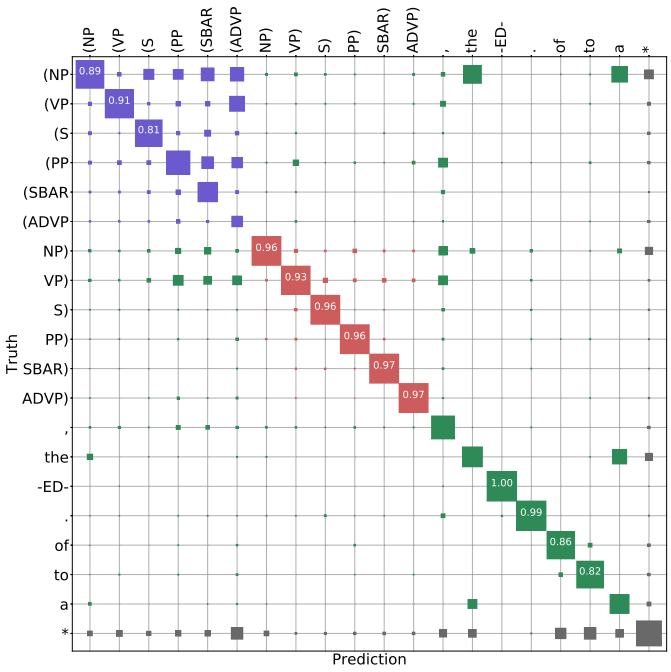

図 10 (4) のテストデータに対する言語モデルの予測結果.

関係を示す．1000 次元ある文脈ベクトルの各要素とネストの深さの相関係数の最大値は 0.92 であった．また，VP の中に入っているかどうかで二値分類した場合のアクティベーションのヒストグラム重複率は，最小で 0.12 であった．図 5 右下と同様，アクティベーションのヒストグラムが規則的にピークを持つことがわかる．これは，および文脈ベクトルの更新が更新ベクトルの加算で行われること，およびその更新ベクトルの各要素が tanh 関数など非線形関数の影響により ± 1 や $\pm \tanh(1) = 0.76..$ に近い値を取ることが多いことにより，文脈ベクトルがその倍数に離散的な値を取ることが原因と考えられる．

また，各要素でなく，要素の線形和で NP, VP などのネストの深さをエンコードしていると仮定したときの結果を示す．以下は VP 句の場合で示すが，NP など他の代表的な句についても同様の傾向となっている．

図 12 に，VP のネストの深さに対する，文脈ベクトルからの線形回帰を行った結果を示す．通常の線形回帰で，相関係数は 0.997 という非常に高いものになった．ペナルティ係数が 0.1 のときの Lasso 回帰を行った場合でも，0.991 となった．このときの非ゼロの要素数は 33 個 (割合は 3.3%) であった．グラフからも，Lasso 回帰と線形回帰の場合に，ほぼ直線上に乗っていることが見て取れる．

以上から，データセット (4) における句のネストの深さは，学習された文脈ベクトルの比較的少数の要素の線形和で表現されており，しかも線形関係にあるといえる．LSTM の構造の観点からは，文脈ベクトルの更新量が単純な加算であることが影響していると考えられる．

最後に，表 2 に，データセット (2) のときと同様，VP 句

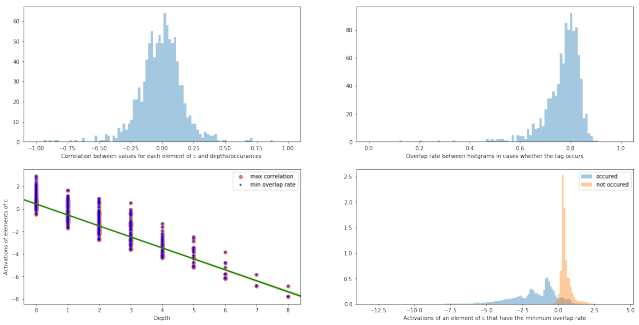

図 11 動詞句 (VP) のネストの深さと各要素のアクティベーション.

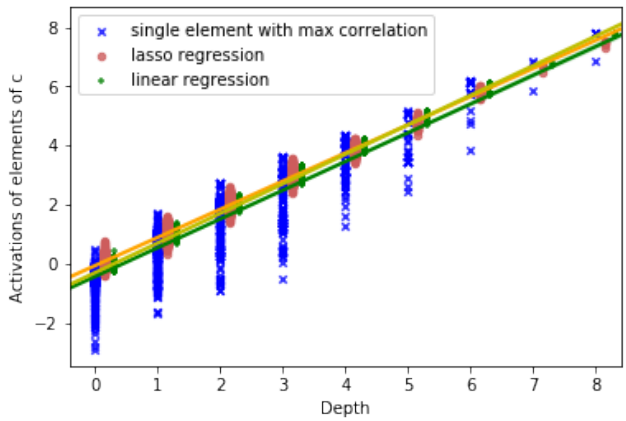

図 12 データセット (4) に対する，VP のネストの深さに対する関数としての Lasso 回帰, 線形回帰, および相関係数が最大となる要素のアクティベーションの比較．スケールは揃えている.

表 2 ロジスティック回帰による VP 内外判定 (データセット (4)).

C	正解率	非ゼロ要素数 (1000 次元中)	非ゼロの割合
0.003	0.9996	134	13%
0.001	0.9992	100	10%
0.0003	0.998	71	7%
0.0001	0.991	51	5%
0.00003	0.97	27	2.7%
0.00001	0.87	12	1.2%

の内側か外側かで判別を行ったときの正解率を示す．チャンスレベルは 0.70 である．ある程度非ゼロ要素数が減少しても，非常に高い精度で分類できていることがわかる．例えば非ゼロの割合が 5% でも，正解率は 0.99 となっている．一方で，非ゼロの割合が 1.2% まで減少すると，正解率が下がっている (0.87).

以上のことから，VP 句のネストの深さや有無の情報は，文脈ベクトルのどれかひとつの要素で完全に表現されるように学習されるというわけではないということ，またその一方で，比較的少数の要素の線形和で表現されるよう学習されると考えることが自然であることがわかる．

7. 関連研究

LSTM がどのように長期依存を捉えるかについての研究としては，Tomita 言語 [13] や簡単な人工言語を用いて 1

または数次元の LSTM での挙動などが過去に調べられている [10][11]．近年 LSTM が様々なタスクに応用されるようになってきたことに伴い，多数の次元を持つ LSTM が構文や長期依存関係をどの程度，どのように捉えているかについて，様々な観点から新たに研究がなされるようになってきている [1][2][7][8]．例えば，Linzen ら [8] は，構文を LSTM を用いた言語モデルが本当に捉えられているかどうかを調べるため，英語における主語と動詞の数の一致 (いわゆる三単現の s) を予測できるかどうかという特定のタスクを設定している．その上で，その正解率に基づき，主語に係る関係節などの構文の複雑になったとき，主語と動詞の数がどの程度予測できなくなるかを調べている．また，Avcu ら [2] は長距離依存関係の複雑さをコントロールできる，正規言語の部分クラスの階層を用いて，LSTM が長期依存関係をどの程度の複雑さまで学習できるかを議論している．また，LSTM や GRU を含めた RNN による言語モデルの学習全般について，理論的にその学習能力の可能性と限界を理解しようとする試みも存在する [3][15]．

8. 議論

本稿では構文木の情報を線形化して与えることで，LSTM が内部でどのように構文情報を獲得しているのか分析した．句構造を陽に与えた場合には，句のネストの深さに対応する要素を LSTM が獲得していることが確認できた．タグ情報を除去した場合にも，句のネストに対応する要素は獲得できている．しかし，今回の調査ではタグ情報を与えない場合でもそれに該当する要素が獲得できているかは確認できていない．タグに該当する要素を LSTM が獲得できるのかや，実際に平文のみから LSTM を用いて言語モデルを学習した場合に，どの程度構文情報を獲得できるのか，文脈ベクトルを含めた内部表現の分析を通じて今後明らかにしていく予定である．

参考文献

[1] Adi, Y., Kermany, E., Belikov, Y., Lavi, O. and Goldberg, Y.: Fine-graind Analysis of Sentence Embeddings Using Auxiliary Prediction Tasks, *Proceedings of the 5th International Conference on Learning Representations (ICLR 2017)* (2017).

[2] Avcu, E., Shibata, C. and Heinz, J.: Subregular Complexity and Deep Learning, *Proceedings of the Conference on Logic and Machine Learning in Natural Language (LaML 2017)*, pp. 20–32 (2017).

[3] Chen, Y., Gilroy, S., Maletti, A., May, J. and Knight, K.: Recurrent Neural Networks as Weighted Language Recognizers, *Proceedings of NAACL-HLT*, pp. 2261–2271 (2018).

[4] Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H. and Bengio, Y.: Learning Phrase Representations using RNN EncoderDecoder for Statistical Machine Translation, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734 (2014).

[5] Hochreiter, S. and Schmidhuber, J.: Long Short-term Memory, *Neural Computation*, Vol. 9.

[6] Khandelwal, U., He, H., Qi, P. and Jurafsky, D.: Sharp Nearby, Fuzzy Far Away: How Neural Language Models Use Context, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pp. 1–11 (2018).

[7] Li, J., Chen, X., Hovy, E. and Jurafsky, D.: Visualizing and Understanding Neural Models in NLP, *Proceedings of NAACL-HLT*, pp. 681–691 (2016).

[8] Linzen, T., Dupoux, E. and Goldberg, Y.: Assessing the Ability of LSTMs to Learn Syntax-Sensitive Dependencies, *Transactions of the Association for Computational Linguistics*, Vol. 4, pp. 521–535 (2016).

[9] Marcus, M., Kim, G., Marcinkiewicz, M. A., MacIntyre, R., Bies, A., Ferguson, M., Katz, K. and Schasberger, B.: The Penn Treebank: Annotating Predicate Argument Structure, *Proceedings of the Workshop on Human Language Technology*, HLT '94, Association for Computational Linguistics, pp. 114–119 (1994).

[10] Prez-Ortiz, J. A., Gers, F. A., Eck, D. and Schmidhuber, J.: Kalman filters improve LSTM network performance in problems unsolvable by traditional recurrent nets, *Neural Networks*, Vol. 16, pp. 241–250 (2003).

[11] Schmidhuber, J.: Deep learning in neural networks: An overview, *Neural Networks*, Vol. 61, pp. 85–117 (2015).

[12] Taylor, A., Marcus, M. and Santorini, B.: *The Penn Treebank: An Overview*, In Treebanks. Text, Speech and Language Technology, Vol. 20, Springer (2003).

[13] Tomita, M.: *Learning of Construction of Finite Automata from Examples Using Hill-climbing: RR: Regular Set Recognizer*, Carnegie-Mellon University, Department of Computer Science (1982).

[14] Weiss, G., Goldberg, Y. and Yahav, E.: Extracting Automata from Recurrent Neural Networks, *Proceedings of the 35th International Conference on Machine Learning, PMLR*, Vol. 80.

[15] Weiss, G., Goldberg, Y. and Yahav, E.: On the Practical Computational Power of Finite Precision RNNs, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pp. 740–745 (2018).