文頭・文末予測の組み合わせによる文特定

宇田川 拓真　　金山 博　　吉田 一星
日本アイ・ビー・エム株式会社 東京基礎研究所
Takuma.Udagawa@ibm.com　{hkana,issei}@jp.ibm.com

概要

文分割とはテキストを連続する文の単位に分割する前処理を指す. 本稿ではこれを拡張し, テキスト中の文でないノイズを除去しつつ, 解析対象とすべき文のみを抽出する文特定タスクの定式化を行う. また, 文頭・文末予測を組み合わせることで文特定を行う新たな手法を提案する. 最後に, Universal Dependencies (UD) のアノテーションに基づく言語横断的なベンチマーク手法を設計し, UD の英語コーパス English Web Treebank (EWT) を用いた実験により提案手法の精度・有用性を検証する.

1　はじめに

自然言語処理において文は重要な処理単位の一つである [1, 2, 5]. 一般的に, テキストはまず文末予測に基づく**文分割**という前処理によって連続する文の単位 (**文単位**と呼ぶ) に分割される [4, 3, 11]. しかし, 実際のテキストデータは必ずしも綺麗な文単位のみを含むとは限らない. 例えば, ウェブデータはメタ情報・文断片・非言語的記号など文と見なせないノイズ（**非文単位**と呼ぶ）を含み得る. 従来の文分割では非文単位もいずれかの文単位に属すると仮定されるが, これらのノイズは構文解析など下流の処理やその評価において障害となることがある [12].

そこで本稿では文分割の代替として, テキストを連続する文単位と非文単位に分割する**文特定**タスクの定式化を行う. それと同時に, 文頭と文末予測を組み合わせることで文特定を行う新たな手法を提案する. 表 1 に示すように, 提案手法では文頭・文末予測によって文単位を特定しつつ, それ以外を非文単位と見なして区別・除去することができる.

次に, 文特定タスクにそのまま適用可能なベンチマークが存在しないため, Universal Dependencies (UD) [8] のアノテーションを基に文特定のベンチマークを設計する. 具体的には, UD が提供する文境界と述語項構造を活用して, 任意の UD コーパスから文特定の正解データを作成する手法を紹介する. 文単位・非文単位の分類基準を目的に応じて柔軟に調整可能であることもこの手法の特長である.

最後に, UD の英語コーパスとして広く使われている English Web Treebank (EWT) [10] を用いて実験を行う. これにより, 提案手法は高い精度で文単位を特定できること, また正確な文単位の特定が構文解析等の下流タスクにおいて有用であることを示す.

2　タスク設計・手法

本節ではまず, 文分割のタスク設計と手法を再定義する. 次に, 文分割の考えを拡張することで文特定のタスク設計と手法を導出できることを示す.

2.1　文分割

入力テキストを $\boldsymbol{W} = (w_0, w_1, ..., w_{N-1})$ とし, 各 w_i は単語とする（文字・サブワードでも可）. また, テキストのスパン $\boldsymbol{W}[i:j] = (w_i, ..., w_{j-1})$ およびその連結 $\boldsymbol{W}[i:j] \oplus \boldsymbol{W}[j:k] = \boldsymbol{W}[i:k]$ を定義する.

文分割では入力テキストを連続する M 個の文単位に分割する.[1] つまり, 文単位の境界のインデックスを $\boldsymbol{B} = (b_0, b_1, ..., b_M)$（ただし $b_0 = 0, b_M = N$）とすると, $\bigoplus_{i=1}^{M} \boldsymbol{W}[b_{i-1}:b_i] = \boldsymbol{W}$ となる. ここで, スパン $\boldsymbol{W}[i:j]$ が一つの文単位となる確率を $p_{\mathrm{SU}}(\boldsymbol{W}[i:j])$ とおくと, 最適な文分割は以下の解 $\boldsymbol{B}$ を求めるタスクとして定式化できる：

$$\arg\max_{\boldsymbol{B}} \prod_{i=1}^{M} p_{\mathrm{SU}}(\boldsymbol{W}[b_{i-1}:b_i]) \tag{1}$$

次に, w_i が文末となる確率のモデル $p_{\mathrm{EOS}}(w_i|\boldsymbol{W};\theta)$ を導入する.[2] 通常, モデルのパラメーター θ は文末（文境界）の正解データを用いて事前学習され [11], このモデルを用いて文単位の確率 $p_{\mathrm{SU}}(\boldsymbol{W}[i:j])$ を以下のように定義できる：

$$p_{\mathrm{SU}}(\boldsymbol{W}[i:j]) = p_{\mathrm{EOS}}(w_{j-1}) \prod_{i \le k < j-1} (1 - p_{\mathrm{EOS}}(w_k)) \tag{2}$$

1）M は変数であることに注意.
2）以降 $\boldsymbol{W}, \theta$ を省略して $p_{\mathrm{EOS}}(w_i)$ のように表記する.

表 1　文分割と文特定の比較. 文分割では文末予測 (**E**) によってテキストを連続する文単位 (青いスパン) に分割する. 文特定では文頭予測 (**B**) と文末予測 (**E**) の間のスパンを文単位として特定し, それ以外のスパンを非文単位と見なす.

入力テキスト (EWT の例)	`Thank you. - TEXT.htm << File: TEXT.htm >> I was thinking of converting it to a hover vehicle. I might just sell the car and get you to drive me around all winter.`
文分割	**E** `Thank you.` **E** `- TEXT.htm << File: TEXT.htm >> I was thinking of converting it to a hover vehicle.` **E** `I might just sell the car and get you to drive me around all winter.`
文特定	**B** **E** `Thank you.` `- TEXT.htm << File: TEXT.htm >>` **B** **E** `I was thinking of converting it to a hover vehicle.` **B** **E** `I might just sell the car and get you to drive me around all winter.`

つまり $\boldsymbol{W}[i:j]$ が文単位となる確率は, そこに含まれる単語のうち最後の単語 w_{j-1} のみが文末となる確率に等しい. この式 (2) を式 (1) に代入すると,

$$\begin{aligned}(1) &= \arg\max_{\boldsymbol{B}} \sum_{i=1}^{M} \left\{ \log p_{\mathrm{EOS}}(w_{b_i-1}) + \sum_{b_{i-1}\le j<b_i-1} \log\left(1-p_{\mathrm{EOS}}(w_j)\right)\right\} \\ &= \arg\max_{\boldsymbol{B}} \sum_{i\in\boldsymbol{B}_{\mathrm{EOS}}} \log p_{\mathrm{EOS}}(w_i) + \sum_{i\notin\boldsymbol{B}_{\mathrm{EOS}}} \log\left(1-p_{\mathrm{EOS}}(w_i)\right)\end{aligned} \tag{3}$$

ここで $\boldsymbol{B}_{\mathrm{EOS}} = \{b_i - 1 \mid i \in (1,2,...,M)\}$ は $\boldsymbol{B}$ が定義する文末のインデックスを指す. 式 (3) は自明な最適化問題であり, $\boldsymbol{B}_{\mathrm{EOS}} = \{i \in (0,1,...,N-1) \mid p_{\mathrm{EOS}}(w_i) \ge 0.5\}$ となる $\boldsymbol{B}$ が最適解となる.

2.2　文特定

文特定では入力テキストを連続する文単位または非文単位に分割する. そのため, a_i を $\boldsymbol{W}[b_{i-1}:b_i]$ が文単位の場合 1, 非文単位の場合 0 を取る変数とすると, 最適な文特定は以下の解 $\boldsymbol{A} = (a_1, a_2, ..., a_M)$ および $\boldsymbol{B}$ を求めるタスクとして定式化できる：

$$\arg\max_{\boldsymbol{B},\boldsymbol{A}} \prod_{i=1}^{M} p_{\mathrm{SU}}(\boldsymbol{W}[b_{i-1}:b_i])^{a_i}\, p_{\mathrm{NSU}}(\boldsymbol{W}[b_{i-1}:b_i])^{1-a_i} \tag{4}$$

ここで $p_{\mathrm{SU}}(\boldsymbol{W}[i:j])$, $p_{\mathrm{NSU}}(\boldsymbol{W}[i:j])$ はそれぞれ $\boldsymbol{W}[i:j]$ が一つの文単位, 非文単位となる確率を指す. 前項の考えを拡張すると, これらの確率は事前学習された文頭確率のモデル $p_{\mathrm{BOS}}(w_i)$ と文末確率のモデル $p_{\mathrm{EOS}}(w_i)$ を用いて以下のように定義できる：

$$\begin{aligned}p_{\mathrm{SU}}(\boldsymbol{W}[i:j]) = {}& p_{\mathrm{BOS}}(w_i) \prod_{i<k\le j-1} \left(1-p_{\mathrm{BOS}}(w_k)\right) \\ & \times p_{\mathrm{EOS}}(w_{j-1}) \prod_{i\le k<j-1} \left(1-p_{\mathrm{EOS}}(w_k)\right) \\ p_{\mathrm{NSU}}(\boldsymbol{W}[i:j]) = {}& \prod_{i\le k\le j-1} \left(1-p_{\mathrm{BOS}}(w_k)\right) \times \prod_{i\le k\le j-1} \left(1-p_{\mathrm{EOS}}(w_k)\right)\end{aligned} \tag{5}$$

つまり $\boldsymbol{W}[i:j]$ が文単位となる確率は, 最初の単語 w_i のみが文頭となり, 最後の単語 w_{j-1} のみが文末となる確率に等しい. また, $\boldsymbol{W}[i:j]$ が非文単位となる確率は文頭および文末が一切含まれない確率に等しい.[3] これらの式 (5) を式 (4) に代入すると,

$$\begin{aligned}(4) = \arg\max_{\boldsymbol{B},\boldsymbol{A}} \sum_{i=1}^{M} \Big\{ & a_i \log p_{\mathrm{SU}}(\boldsymbol{W}[b_{i-1}:b_i]) \\ & + (1-a_i) \log p_{\mathrm{NSU}}(\boldsymbol{W}[b_{i-1}:b_i]) \Big\} \\ = \arg\max_{\boldsymbol{B},\boldsymbol{A}} \sum_{i\in\boldsymbol{B}^{\boldsymbol{A}}_{\mathrm{BOS}}} & \log p_{\mathrm{BOS}}(w_i) + \sum_{i\notin\boldsymbol{B}^{\boldsymbol{A}}_{\mathrm{BOS}}} \log\left(1-p_{\mathrm{BOS}}(w_i)\right) \\ & + \sum_{i\in\boldsymbol{B}^{\boldsymbol{A}}_{\mathrm{EOS}}} \log p_{\mathrm{EOS}}(w_i) + \sum_{i\notin\boldsymbol{B}^{\boldsymbol{A}}_{\mathrm{EOS}}} \log\left(1-p_{\mathrm{EOS}}(w_i)\right)\end{aligned} \tag{6}$$

ここで $\boldsymbol{B}^{\boldsymbol{A}}_{\mathrm{BOS}} = \{b_{i-1} \mid i \in (1,2,...,M), a_i = 1\}$ は文頭のインデックス, $\boldsymbol{B}^{\boldsymbol{A}}_{\mathrm{EOS}} = \{b_i - 1 \mid i \in (1,2,...,M), a_i = 1\}$ は文末のインデックスを指す. 付録 A に示すように, 式 (6) の最適解 $\boldsymbol{B}, \boldsymbol{A}$ は動的計画法を用いて厳密かつ効率的に求めることができる.

3　評価

本節では, 任意の言語の UD コーパスから文特定のベンチマークを構築する手法について説明する. 具体的には UD のアノテーションを基に, まず (1) 文単位・非文単位の境界を決定し, 次に (2) 各区分を文単位または非文単位に分類する.

まず手順 (1) では, UD における文境界[4] を文単位・非文単位の境界として採用する. 手順 (2) においては, 言語学における Lexical Sentence [9] の考え方を採用し, 単語間の依存関係を基に文単位を定義する. 具

3)　この定義に基づくと, 任意の $\boldsymbol{W}[i:j] \oplus \boldsymbol{W}[j:k] = \boldsymbol{W}[i:k]$ に対して $p_{\mathrm{NSU}}(\boldsymbol{W}[i:k]) = p_{\mathrm{NSU}}(\boldsymbol{W}[i:j]) \times p_{\mathrm{NSU}}(\boldsymbol{W}[j:k])$ が成立する. つまり, 連続する非文単位の区別は行われない.

4)　UD において, 文境界における「文」とは表 2 のような非文単位を含む広義の意味で使われている.

表 2　English Web Treebank (EWT) に含まれる非文単位の代表例. 各行は一つの非文単位に対応する.

~~*~*~*~*~*~*~*~*
**********NOTE**********
By video conference from _____
Excerpt:
02/13/2001 08:02 PM
5:00 PT ** 6:00 MT ** 7:00 CT ** 8:00 ET
Time: 11:30am / 1:30pm Central / 2:30pm Eastern
Sunshine Coast, British Columbia, Canada
- UnleadedStocks.pdf
t r u t h o u t — Perspective
(Answered, 2 Comments)
The federal sites of Washington, DC.

表 3　EWT ベンチマークの統計値.

	訓練	検証	テスト
文単位	10,356	1,523	1,490
非文単位	2,187	478	587
B ラベル	10,356	1,523	1,490
I ラベル	160,127	18,791	18,222
O ラベル	6,939	1,302	1,822

体的には, **節**を文単位を構成する重要な要素と考え, 節を成す述語とそれに係る依存関係（core argument または non-core dependent）が少なくとも一つ含まれる区分を文単位として分類する.[5)]

本稿では実験のため, English Web Treebank (EWT) [10] を用いて実際にベンチマークを構築する. EWT は E メール, 製品レビュー, QA サイトなどウェブ上の様々なジャンルのテキストデータから成り立ち, 解析対象とすべき文単位（表 6）だけでなく多様な非文単位（表 2）が含まれる. なお, 本稿の基準では独立した名詞句等も非文単位に分類されるが, 目的に応じてこの基準は柔軟に調整可能である.

また, EWT ベンチマークの統計値を表 3 に示す. 本稿の基準では, 約 17~28% と多くの区分が非文単位として分類されることが分かる. 一方で, 文単位を単語レベルの BIO ラベル[6)]を用いてアノテーションした場合, 約 4~8%の単語のみが O ラベルを付与される. これは表 2 に示すように, 非文単位が比較的少ない単語数で構成されることに起因する.

4　実験

最後に, EWT ベンチマークを用いた実験により文特定の精度と有用性を検証する.

5）UD の依存関係の詳細については以下を参照されたい：https://universaldependencies.org/u/dep/.

6）B は文単位の先頭, I は文単位の途中, O は文単位の外（つまり非文単位）であることを示す.

4.1　実験設定

2 節で述べたように, 文特定を行うには事前学習された文頭確率と文末確率のモデルが必要である. そのため, 本稿では RoBERTa [6] をベースに文頭と文末を予測する分類モデルを構築する.

モデルの訓練・検証時には, 入力テキストとして連続する L 個の文単位・非文単位を連結する. ここで L は成功確率 $p_{cc} = 0.5$ の幾何分布からサンプリングする.[7)] ただし, 入力長が RoBERTa の上限を超える場合, 上限を超えない最大の $L' < L$ を代用する. この手順により, 様々な長さの入力テキストに対応するモデルが構築できると期待される.

テスト時には, より多様な入力テキストを考慮して以下の設定を考える. 第一に, 訓練・検証時と同様に連続する文単位・非文単位を連結する方法が考えられる. 本稿では $p_{cc} = 0.5$ と, 可能な限り長い入力を想定した $p_{cc} = 0$ を用いて評価を行う.

第二に, 文分割の結果の後処理として文特定を行う設定を考える. 具体的には, 高性能な文分割モデル [11] を用いて EWT の生データを分割し, その結果に対して文特定を行う. なお, 文分割の段階で文単位が誤って分割されてしまった場合は, 分割された部分を非文単位と見なして正解を修正する.

最後に, 文特定の精度を文単位の抽出タスクとして評価する. 具体的には, 文単位の単語レベルの BIO ラベル (3 節参照) の F1 スコアと, スパンレベルの完全一致の F1 スコアを用いる. また, ベースラインとして文末確率のモデルのみを用いた文分割手法（2.1 項）との比較を行う. 文末確率のモデルは文特定と同様に訓練し, 入力の最後の単語を強制的に文末とする場合・しない場合の両方で評価する.[8)]

4.2　実験結果

表 4 に文単位の抽出タスクの実験結果を示す.

第一に, 提案手法（文特定）は全ての評価指標において文分割を大きく上回ることが確認できる. 特にスパンレベルの完全一致でも, 提案手法は 84~89% と高い精度を達成している. また, 長い入力（$p_{cc} = 0$）に対しては文分割では大きな性能劣化が

7）$p_{cc} \in (0, 1]$ の時, 幾何分布の確率質量関数は $l \in \mathbb{N}$ において $p(L = l) = (1 - p_{cc})^{l-1} p_{cc}$ となり, p_{cc} が小さくなるほど L は大きくなりやすい. $p_{cc} = 0$ の時, $p(L = \infty) = 1$ とみなす.

8）最後の単語を強制的に文末とする場合, 非文単位は一切予測されない. これを強制しない場合, 予測された最後の文末以降のスパンを（存在すれば）非文単位と見なす.

表 4　文単位の抽出タスクの実験結果. 単語レベルの BIO ラベルとスパンレベルの F1 スコアによって評価する.

手法	EWT テスト (p_{cc} = 0.5)				EWT テスト (p_{cc} = 0)				EWT テスト (後処理)			
	B	I	O	Span	B	I	O	Span	B	I	O	Span
文末による文分割	85.6	97.3	66.6	72.8	78.0	95.1	6.0	58.2	90.2	97.5	71.3	81.6
文末による文分割 (＋文末強制)	79.8	95.9	0.0	60.4	77.8	95.1	0.0	57.7	81.7	95.7	0.0	62.3
文頭・文末による文特定	**94.3**	**98.7**	**86.1**	**87.3**	**93.0**	**98.2**	**81.7**	**84.1**	**94.7**	**98.4**	**83.9**	**88.8**

表 5　下流タスク (単語区切り/ Words・品詞タグ付け/ UPOS・依存構造解析/ LAS) の評価結果.

EWT 訓練・検証データ	EWT テストデータ								
	全体			文単位のみ			非文単位のみ		
	Words	UPOS	LAS	Words	UPOS	LAS	Words	UPOS	LAS
全体	98.6	96.2	89.7	98.8	96.5	92.0	96.5	92.0	81.4
文単位のみ	-	95.7	89.0	-	96.5	90.2	-	88.3	75.4
非文単位のみ	-	72.6	36.1	-	70.7	31.8	-	91.0	77.1

見られる[9]一方で, 文特定の手法ではこれらに対しても高い性能を維持していることが分かる.

以上の結果より, 非文単位を除いて文単位を特定するには文末の予測のみ（= 文分割）では不十分であり, 文頭・文末の予測を組み合わせること（= 文特定）が重要であると言える.

4.3　下流タスクの評価

最後に, 文特定の下流タスクにおける有用性について, 単語区切り・品詞タグ付け・依存構造解析の３つに焦点を当てて検証する.

まず背景として, EWT は非文単位のようなノイズを含むため, 解析器の訓練・評価の両面で支障があることが知られている [12]. そこで本稿では, 理想的な文単位・非文単位の区分[10]に基づいて解析器を訓練・評価することの利点について調査する.

下流タスクの解析器には Trankit [7] を用いる. 手順としては UD-EWT v2.6 の全体 (文単位と非文単位) を用いて単語区切りを訓練し[11], 続いて UD-EWT v2.10 の全体・文単位のみ・非文単位のみのそれぞれで品詞タグ付けと依存構造解析を訓練する. 最後に, UD-EWT v2.10 の全体・文単位のみ・非文単位のみのそれぞれで各解析器を評価する.

表 5 に下流タスクの評価結果を示す. まず EWT 全体で解析器を訓練した場合, 全ての下流タスクにおいて文単位のみで評価したスコアが高く, 非文単位のみで評価したスコアが顕著に低くなっている. このことから, 文単位は予測に適した綺麗なデータが多く, 逆に非文単位は予測しづらいイレギュラーなデータが多いことが分かる. これらを同一視せずに区別して評価を行うことは, 解析器の特徴を正しく理解するために重要と考えられる.

また, 文単位のみで解析器を訓練した場合, 学習データは少なくなるにもかかわらず, 全体で訓練した場合と近い性能が出ることが分かる.[12] 例えば文単位に対する品詞タグ付けでは, 全体で訓練した場合と同等の性能が出ている. 逆に非文単位のみで訓練した場合の性能は全体的に低く, 訓練データとしても有用でないノイズが多いと考えられる.[13] 今後の応用として, 例えば文単位・非文単位の分類基準（3 節）を調整することで訓練に最適なデータのフィルタリングを行えると考えられる.

以上の結果より, 正確な文特定は解析器の訓練・評価の両面で有用であると考えられる.

5　まとめ

本稿では文分割の考え方を拡張し, 非文単位を除きつつ文単位を特定する文特定のタスクと手法を提案した. また, EWT を用いた実験により文特定の精度と有用性を確認した. 今後の課題として, 文特定のさらなる精度向上, 異なる言語・ドメインにおける有効性の検証, 実際の（誤りを含む）予測に基づく下流タスクの評価・分析などが挙げられる.

9)　文分割では予測できる非文単位 (O ラベル) の割合が入力が長くなるほど小さくなり, 結果的に性能が大幅に劣化する.
10)　つまり 3 節で示した基準で分類した正解の区分を用いる. 現実的には文特定の予測の段階で誤りが生じる可能性があるが, その点まで考慮した分析は今後の課題とする.
11)　複数語トークンによるエラーを回避するため, それを持たない旧バージョンで訓練した単語区切りモデルで固定した.
12)　ただし非文単位のみで評価した結果は異なる傾向を持ち, やはりイレギュラーなデータを多く含むことを示唆している.
13)　フェアな比較のためには訓練に使うデータ量を揃える必要があるが, 付録 C に示すようにこの設定でも同様の傾向が確認された.

参考文献

[1] Timothy Dozat and Christopher D Manning. Deep biaffine attention for neural dependency parsing. In *Proc. of ICLR*, 2017.

[2] Timothy Dozat and Christopher D. Manning. Simpler but more accurate semantic dependency parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 484–490, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[3] Dan Gillick. Sentence boundary detection and the problem with the U.S. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pp. 241–244, Boulder, Colorado, June 2009. Association for Computational Linguistics.

[4] Tibor Kiss and Jan Strunk. Unsupervised multilingual sentence boundary detection. *Computational Linguistics*, Vol. 32, No. 4, pp. 485–525, 2006.

[5] Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*, Vol. 8, pp. 726–742, 2020.

[6] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[7] Minh Van Nguyen, Viet Dac Lai, Amir Pouran Ben Veyseh, and Thien Huu Nguyen. Trankit: A light-weight transformer-based toolkit for multilingual natural language processing. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pp. 80–90, Online, April 2021. Association for Computational Linguistics.

[8] Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Jan Hajič, Christopher D. Manning, Sampo Pyysalo, Sebastian Schuster, Francis Tyers, and Daniel Zeman. Universal Dependencies v2: An evergrowing multilingual treebank collection. In *Proceedings of the 12th Language Resources and Evaluation Conference (LREC 2020)*, pp. 4034–4043, Marseille, France, May 2020. European Language Resources Association.

[9] Geoffrey Nunberg. *The linguistics of punctuation*. No. 18. Center for the Study of Language (CSLI), 1990.

[10] Natalia Silveira, Timothy Dozat, Marie-Catherine de Marneffe, Samuel Bowman, Miriam Connor, John Bauer, and Chris Manning. A gold standard dependency corpus for English. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pp. 2897–2904, Reykjavik, Iceland, May 2014. European Language Resources Association (ELRA).

[11] Rachel Wicks and Matt Post. A unified approach to sentence segmentation of punctuated text in many languages. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 3995–4007, Online, August 2021. Association for Computational Linguistics.

[12] 金山博, 大湖卓也. UD_English-EWT との付き合い方. 言語処理学会第 28 回年次大会予稿集, March 2022.

表 6　English Web Treebank (EWT) に含まれる文単位の代表例. 各行は一つの文単位に対応する.

President Bush on Tuesday nominated two individuals to replace retiring jurists on federal courts in the Washington area.
Unfortunately, Mr. Lay will be in San Jose, CA participating in a conference, where he is a speaker, on June 14.
“In 1972, there was an enormous glut of pilots,” Campenni says.
PS – There is a happy hour tonight at Scudeiros on Dallas Street (just west of the Met Garage) beginning around 5:00.
2) Your vet would not prescribe them if they didn’t think it would be helpful.
BUT EVERYONE HAS THERE OWN WAY!!!!!!
The motel is very well maintained, and the managers are so accomodating, it’s kind of like visiting family each year! ;-)
where can I find the best tours to the Mekong Delta at reasonable prices?

表 7　訓練データ量を揃えた場合の下流タスク (単語区切り・品詞タグ付け・依存構造解析) の評価結果.

EWT 訓練・検証データ	EWT テストデータ								
	全体			文単位のみ			非文単位のみ		
	Words	UPOS	LAS	Words	UPOS	LAS	Words	UPOS	LAS
全体	98.6	94.6	82.8	98.8	95.1	83.7	96.5	90.2	71.4
文単位のみ	-	94.0	82.4	-	94.9	83.9	-	85.5	65.3
非文単位のみ	-	72.6	36.1	-	70.7	31.8	-	91.0	77.1

A　動的計画法

2.2 項の式 (6) の最適解は動的計画法を用いて求めることができる. 具体的には, 任意の $k \leq N-1$ に対して $\boldsymbol{W}^{\leq k} = (w_0, ..., w_k)$ までの最適な文頭・文末予測を考える. つまり, 式 (6) における $\boldsymbol{W}$ を $\boldsymbol{W}^{\leq k}$ に置き換えた最適化問題・目的関数を考える.

$\boldsymbol{W}^{\leq k}$ までの予測は不完全であるため, 文単位の途中で終わる（= 最後の予測が文頭である）場合と, 文単位の外で終わる（= 最後の予測が文頭でない）場合の 2 通りが考えられる. $\log p_{\mathrm{IS}}(k+1)$ を前者の場合の目的関数の最大値, $\log p_{\mathrm{OS}}(k+1)$ を後者の場合の目的関数の最大値と置く. すると, 予測は必ず文単位の外から始まるので $\log p_{\mathrm{IS}}(0) = \log 0 = -\infty$, $\log p_{\mathrm{OS}}(0) = \log 1 = 0$ と初期化でき, これらの値を以下のように更新して求めることができる.

$$
\begin{aligned}
\log p'_{\mathrm{IS}}(i) &= \max\{\log p_{\mathrm{IS}}(i) + \log(1-p_{\mathrm{BOS}}(w_i)), \\
&\qquad \log p_{\mathrm{OS}}(i) + \log p_{\mathrm{BOS}}(w_i)\} \\
\log p'_{\mathrm{OS}}(i) &= \log p_{\mathrm{OS}}(i) + \log(1-p_{\mathrm{BOS}}(w_i)) \\
\log p_{\mathrm{IS}}(i+1) &= \log p'_{\mathrm{IS}}(i) + \log(1-p_{\mathrm{EOS}}(w_i)) \\
\log p_{\mathrm{OS}}(i+1) &= \max\{\log p'_{\mathrm{IS}}(i) + \log p_{\mathrm{EOS}}(w_i), \\
&\qquad \log p'_{\mathrm{OS}}(i) + \log(1-p_{\mathrm{EOS}}(w_i))\}
\end{aligned}
\tag{7}
$$

この更新式ではまず, 文頭確率 $p_{\mathrm{BOS}}(w_i)$ に基づいて $p_{\mathrm{IS}}(i) \to p'_{\mathrm{IS}}(i)$, $p_{\mathrm{OS}}(i) \to p'_{\mathrm{OS}}(i)$ の更新を行う. 次に, 文末確率 $p_{\mathrm{EOS}}(w_i)$ に基づいて $p'_{\mathrm{IS}}(i) \to p_{\mathrm{IS}}(i+1)$, $p'_{\mathrm{OS}}(i) \to p_{\mathrm{OS}}(i+1)$ の更新を行う.

最終的に予測は文単位の外で終わる必要があるため, $\log p_{\mathrm{OS}}(N)$ が式 (6) の目的関数の最大値となる. 式 (6) の最適解 B, A は, 更新式 (7) をバックトラックすることで簡単に求めることができる.

B　文単位の例

表 6 に EWT に含まれる文単位の代表例を示す. これらの例に示されるように, 本稿の基準では節を成す述語項構造を持つ区分のみが解析対象とすべき文単位として判定される.

C　下流タスク評価の追加結果

表 7 に訓練データ量を揃えた場合の下流タスクの評価結果を示す. 手順としては, 非文単位のデータ量が最も少ないため, 訓練データに含まれる単語数を非文単位におおよそ合わせて訓練を行った.[14)]

この結果から, やはり文単位のみで解析器を訓練した場合, 全体で訓練した場合と同等の性能が出ることが分かる. また, データ量を揃えた場合でも非文単位のみで訓練した場合の性能は全体として低いことが分かる. このことから, 下流タスクの学習には文単位が最も有用であり, 非文単位の貢献は非常に小さいことが分かる. ただし非文単位のみで評価した場合の結果は異なる傾向を示しており, 非文単位に含まれるデータは（アノテーションが一貫していないなど）イレギュラーであることが示唆される.

14)　訓練データの単語数をおおよそ合わせるために, 全体で訓練する場合はランダムに 622 例, 文単位のみで訓練する場合はランダムに 518 例, 非文単位のみで訓練する場合は 2,187 例（全ての例）を使用した.