Vol.2015-NL-223 No.10
2015/9/28

# 複合機能語を考慮した英語の依存構造コーパスの構築

加藤 明彦[†1,a)]　進藤 裕之[†1,b)]　松本 裕治[†1,c)]

**概要**：複単語表現 (MWE) を正しく認識して適切に取り扱う事は, 構文解析や意味解析などの言語解析で重要である. しかし Penn Treebank から変換して得られる依存構造では各単語がノードとなっており, MWE に関する考慮は行われていない. そこで本稿では, 機能表現を担う MWE である複合機能語を考慮した英語の依存構造コーパスを構築した. また, 構築したコーパスを用いた依存構造解析を行った為, その結果も合わせて報告する.

## 1. はじめに

複単語表現 (Multiword Expression, MWE) とは, 単語境界を越えて特異な解釈を持つ表現である [1]. MWE を正しく認識して適切に取り扱う事は, 構文解析, 意味解析などの言語解析で重要である.

言語資源の観点から考えると, MWE を考慮した構文解析を行う為には, 句構造木もしくは依存構造木に MWE の情報 (MWE の範囲や品詞) を取り込んだコーパスが提供されている事が望ましい. 例えば フランス語の句構造木のコーパスである French Treebank [2] では MWE が部分木にまとめられており, MWE 認識と構文解析の双方を対象とした研究 [3], [4] の実験データとして良く用いられている. しかし英語の構文解析で標準的に用いられている PennTreebank において MWE は考慮されていない. また, 単語ベースの依存構造で MWE 配下のノード群を単一ノードにまとめる手法で得られるグラフは, 事例によっては MWE を考慮した依存構造として望ましくない事がある（ 3 章 ）.

そこで我々は句構造木中の MWE を単一の部分木にまとめた上で依存構造に変換する手法を提案する. この手法を適用する事で, MWE としての品詞を考慮し, MWE を 1 ノードとした依存構造木を得る事ができる. 本研究では提案手法を Ontonotes コーパスに適用する事により, MWE の中でも機能表現に相当する複合機能語の情報を反映させた依存構造コーパスを構築した. この際, MWE を部分木にまとめる手順のアノテーションコストを抑える為, 句構造木中の MWE のパターンを修正容易性の観点で分類し, 自動変換が難しいものに限り人手での修正を行った. また, 上記で構築したコーパスに対する依存構造解析を行い, 変換前のコーパスに対する依存構造解析と定量的, 定性的の両面で比較を行った（ 4 章 ）.

[†1] 奈良先端科学技術大学院大学 情報科学研究科
Nara Institute Science of Technology
[a)] kato.akihiko.ju6@is.naist.jp
[b)] shindo@is.naist.jp
[c)] matsu@is.naist.jp

## 2. 関連研究

重藤ら [5] は複合機能語を対象に, 英語の MWE 辞書の作成と Penn Treebank に出現する MWE のアノテーションを行っている. 各 MWE については範囲と品詞がタグ付けされているが, MWE を考慮した依存構造までは提供されていない.

MWE に関する研究の多くは連続 MWE を対象としているが, 句動詞の様に構成要素間にギャップを持つ MWE を対象に含めた研究例も見られる. 例えば Schneider ら [6] は English WebTreebank を対象に, ギャップありの MWE も含め, 様々な品詞の MWE をタグ付けしたコーパスを構築している. ただし MWE を考慮した構文構造までは提供されていない点, コーパスのサイズが約 3800 文と構文解析器の訓練データとするには比較的小さい点が本コーパスとは異なる.

また, MWE を考慮した依存構造コーパスの例として, Universal Dependency [7] が挙げられる. Universal Dependency は言語横断的に一貫した Treebank を提供する事を目的としたプロジェクトであり, 複合機能語については, 構成要素の内, 先頭以外の全てのトークンは先頭トークンを head とし, エッジに “MWE” という dependency label が付与される仕様となっている. つまり MWE の範囲も依存構造の中で表現される. Universal Dependency では MWE の構成要素はそれぞれが依存構造木の独立したノードであり, MWE 全体を 1 ノードとした依存構造を提供す

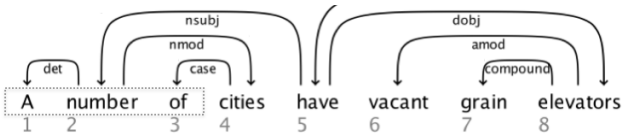


**図 1** MWE(“a number of”) (A) : 変換前の依存構造 (“a number of cities” の syntactic head は “number”)

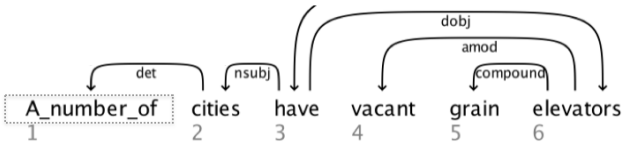


**図 2** MWE(“a number of”) (B) : MWE を考慮した依存構造 (“a number of cities” の syntactic head は “cities”)

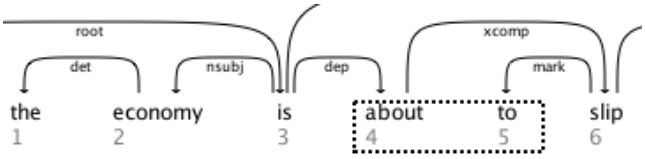


**図 3** MWE 配下のノードを 1 つにまとめると, ループと multi head が生じる例 (“about to”)

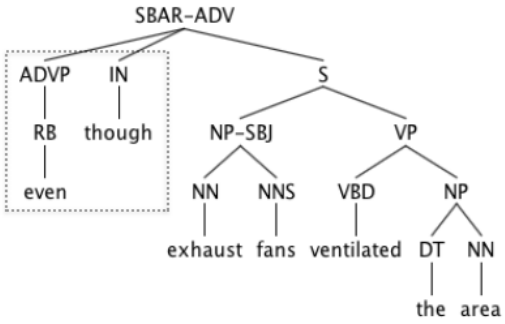


**図 4** Simple (1) : 変換前の LCA-tree. MWE (“even though”) 内外のトークンが混在する部分木を持たない.

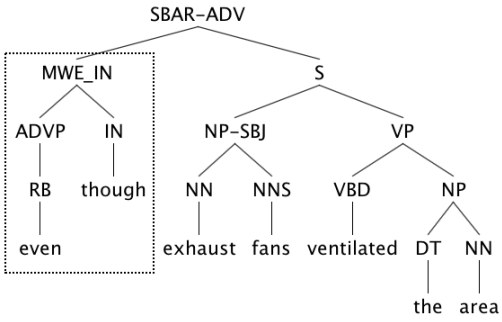


**図 5** Simple (2) : MWE (“even though”) を部分木にまとめた LCA-tree

る我々のコーパスとはこの点で異なっている.

MWE を考慮した構文解析については, MWE 認識を構文解析の前処理, もしくは同時に行った研究が報告されている [3], [4]. また, Nivre ら [8] はスウェーデン語を対象として, MWE を考慮すると, MWE 認識を完璧に行うという理想的な条件の下で依存構造解析の精度が向上する事を報告している. 彼らが対象とした MWE は 複合名詞 (人名/地名), 数値/数式, 複合機能語である.

## 3. 複合機能語を考慮した依存構造コーパスの構築

我々が今回構築するコーパスの元となる重藤ら [5] のコーパスでは, MWE 全体に 1 つの品詞が付与されている. その為, MWE を考慮した依存構造では文中の MWE が 1 つのノードとなっている事が望ましい. 一方, これまで用いられて来た依存構造では各単語がノードになっている. 後者から前者への直接変換を試みた場合, 単語ベースの依存構造において, MWE を構成するノード群を単一のノードにまとめる事になる. しかし, 例えば **図 1** の例でこの手順を適用した場合, “number” → “cities” と “cities” → “of” のエッジによってループが生じる為, 依存構造木を得る為にはいずれかのエッジを取り除く必要がある. この様な場合にどちらのエッジを除去するかは事例ごとに判断する必要がある. また, MWE を考慮した依存構造 ( **図 2** ) では “a number of” は限定詞である為,“a number of cities” の syntactic head は “cities” となる. 従って “have” の dependent は “number” でなく “cities” となるが, これも上記の方法では得る事が出来ない. また **図 3** の例で, MWE の構成トークン群を単一のノードにまとめる方法を適用すると,“about to” は “is” と “slip” という複数の head（multi head）を持ち, かつ,“about to” と “slip” の間でループが生じてしまう.

上記の問題点を解決する為に, 我々はまず句構造木において MWE を単一の部分木にまとめ, その後で依存構造に変換するというアプローチを採用する. このアプローチであれば, 上述したループや multi head の発生を回避する事ができる. ただし MWE が出現する全ての句構造木に対するアノテーションを人手で行うのはコストがかかる為, 句構造木中の MWE のパターンを修正容易性の観点から Simple, Complex の 2 種に分類し（ 3.1）, 自動変換が難しいものに限り人手での修正を行った. 具体的には以下の手順で複合機能語を考慮した依存構造コーパスを作成した.

(1) Ontonotes の句構造木中の MWE を一つの部分木としてまとめる (**図 4** → **図 5**) [*1]

(2) MWE の構成トークンをアンダースコアで連結したノードを子に持つ preterminal(MWE としての品詞を持つ)

[*1] 文中の MWE の位置と, MWE としての品詞については [5] を利用した.

で当該部分木を差し替える ( **図 6** )

(3) Stanford Dependency [9] に変換する [*2]

### 3.1 MWE の部分木へのまとめ上げ

上記のステップ (1) では, 句構造木中の MWE を一つの部分木にまとめる. 例えば 図 4 の “even though” は, [5] で MWE として注釈されている. 我々はこれを図 5 の様に修正する. この様に, MWE を単一の部分木にまとめても他の部分木の構造に影響がないケースを Simple, それ以外を Complex とする.

なお, 句構造木中の MWE を一つの部分木にまとめる際には, MWE 構成トークン群の LCA(Least Common Ancestor, 最近共通祖先) を根とする木 (以下, LCA-tree) を考えれば良い. 例えば図 4 では, “even” と “though” の LCA (SBAR-ADV) を根とする木が LCA-tree である.

#### 3.1.1 Simple

Simple ケースでは MWE の全ての構成トークンが LCA の子ノードもしくは子孫 (リーフノードから LCA に至るパス上の内部ノードは全て分岐数 1) となる. この為, MWE をカバーする部分木群を一つの新しい中間ノードの子とし, この中間ノードを LCA の子とする (図 4 → 図 5).

#### 3.1.2 Complex

Complex ケースでは, 例えば **図 7** → **図 8** の様に MWE を部分木にまとめる事により, 他の部分木の構造に影響が及ぶ (図 7 の LCA の右の子 (PP) は, 図 8 には存在しない).

我々は変換前後で LCA-tree がどの程度保たれるかに基づいて Complex ケースを Complex-normal, Complex-abnormal の 2 種に分類した為, これについて以下で説明する.

##### Complex-normal

このサブケースでは, LCA-tree 内の MWE 以外のトークンが属する部分木を破壊する事なく MWE を一つの部分木としてまとめる事ができる.

例えば図 7 では, LCA の右部分木の様に, MWE 内外のトークンが混在した部分木が存在するが, LCA の孫ノード (NP) の様に, MWE より前や後のスパンを過不足無く覆う内部ノードが存在する.

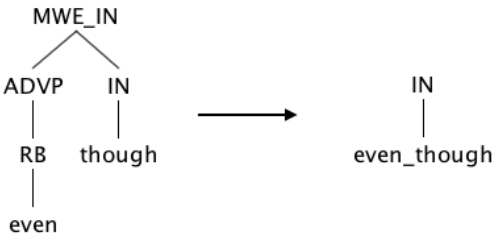


**図 6** MWE (“even though”) をまとめた部分木の差し替え

[*2] 変換コマンドのオプションとしては -conllx -basic -makeCopulaHead -keepPunct を指定した.

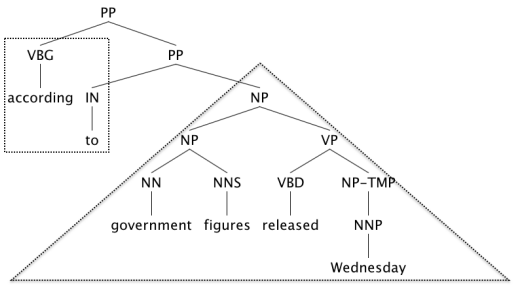


**図 7** Complex-normal (1) : 変換前の LCA-tree. MWE (“according to”) 内外のトークンが混在する部分木を持つ. 図中の三角は, MWE より後のスパンを過不足無く覆う内部ノードを根とする部分木 ($T_{suffix}$) を示す.

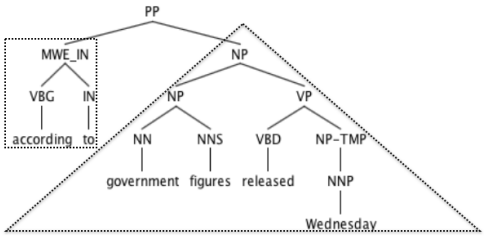


**図 8** Complex-normal (2) : MWE (“according to”) を部分木にまとめた LCA-tree. 図中の三角は $T_{suffix}$ を示す.

ここで, MWE より前のスパンを過不足無く覆う内部ノードを根とする部分木を $T_{prefix}$, MWE をまとめた部分木を $T_{mwe}$, MWE より後のスパンを過不足無く覆う内部ノードを根とする部分木を $T_{suffix}$ とする. 例えば図 7 では, 図中で三角で囲った部分木が $T_{suffix}$ である.

我々は, LCA が $T_{prefix}$, $T_{mwe}$, $T_{suffix}$ を子孫に持つ様に木の変換を行った (図 7 → 図 8). なお, $T_{prefix}$ と $T_{suffix}$ の双方が存在する事例では, $T_{prefix}$, $T_{mwe}$, $T_{suffix}$ をフラットに LCA に持たせるのか, それとも $T_{prefix}$, $T_{suffix}$ のいずれかを先に $T_{mwe}$ とまとめてから, そのノードを LCA に持たせるのか, 人手で判断を行った.

また Simple ケース, Complex-normal ケースについては Ontonotes から導出した PCFG を利用して, (当該事例で LCA の親ノードが LCA を含めた子ノード群を生成する規則の確率) * (LCA が子ノード群を生成する規則の確率) を計算し, 左記の確率値を最大化する LCA の major category (例 : NP-xxx であれば NP) が変換前の LCA と異なる場合, LCA のシンボルを前者で置換した.

##### Complex-abnormal

MWE を一つの部分木としてまとめる際, LCA-tree 内の MWE 以外のトークンが属する部分木を破壊せざるを得な

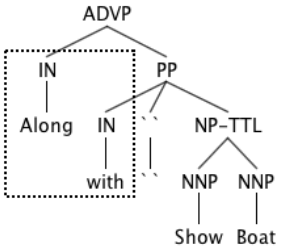


**図 9** Complex-abnormal (1) : 変換前の LCA-tree. MWE(“along with”) より後のスパンを過不足無く覆う内部ノードが LCA-tree 中に存在しない.

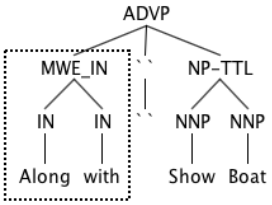


**図 10** Complex-abnormal (2) : MWE(“along with”) を部分木にまとめた LCA-tree

**表 1** 複合機能語を考慮した依存構造コーパス中の各ケースの出現頻度と MWE の種類数

| ケース名 | 事例数 | MWE の種類数 |
|---|---|---|
| Simple | 5129 | 427 |
| Complex-normal | 1742 | 117 |
| Complex-abnormal | 57 | 27 |

いケースである (**図 9** → **図 10**). この場合, LCA-tree のいずれかの部分木で MWE 内外のトークンが混在しており, かつ, MWE より前や後のスパンを過不足無く覆う内部ノードが LCA-tree 中に存在しない. そこで MWE をまとめた部分木と, MWE より前や後のスパンに相当する部分木群をどの様にまとめ上げるかについて事例ごとに判断し, 人手で修正を行った.

なお, 構築したコーパス (Ontonotes の Wall Street Journal のセクション 00-24) 中の各ケースの出現頻度 及び MWE の種類数を**表 1** に示す.

## 4. 複合機能語を考慮した依存構造解析

本章では 3 章の手順で構築したコーパスを用いた依存構造解析について述べる.

### 4.1 実験設定

1 次の MST Parser [10] を用い, 品詞タグは訓練時, テスト時 共にゴールドデータを利用した. 訓練データは Ontonotes(Wall Street Journal) のセクション 02-21, テストデータはセクション 23 である. オリジナルの依存構造と MWE を考慮した依存構造について各々独立に訓練・テストを行い, 前者のテスト時の精度をベースラインとし, 評価指標としては UAS(ラベル無し正解率) を用いた.

**表 2** 各依存構造解析の結果

| | UAS (全体) | UAS (MWE を含む文) |
|---|---|---|
| original | 89.99 | 87.77 |
| mwe-aware | 90.01 | 87.84 |

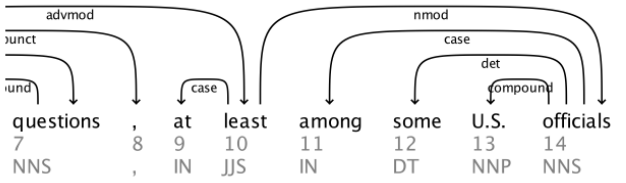


**図 11** オリジナルの依存構造解析器では正解できなかったが, MWE (“at least”) を考慮した依存構造解析器では正解した事例 : (a) オリジナルの解析器により推定された依存構造. “officials” の head として, 正解の “questions” ではなく, MWE(“at least”) の構成トークン (“least”) を 推定している. 全文は “Mrs. Hills’ remarks did raise questions, at least among some U.S. officials, about what exactly her stance is on U.S. access to the Japanese semiconductor market.”

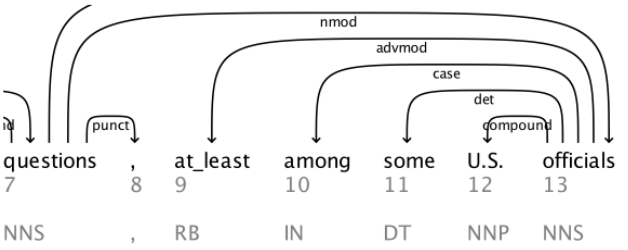


**図 12** オリジナルの依存構造解析器では正解できなかったが, MWE (“at least”) を考慮した依存構造解析器では正解した事例 : (b) MWE を考慮した解析器により推定された依存構造. “officials” の head として, 正解の “questions” を推定している.

### 4.2 実験結果, 考察

実験結果を**表 2** に示す. MWE を考慮した場合の UAS 値は, 全事例 (1640 文), MWE を含む事例 (266 文) のいずれに対してもベースラインからわずかに向上し, 全事例については 0.02 ポイント, MWE を含む事例については 0.07 ポイントの上昇となった.

以下では定性的な分析を行う. まず, オリジナルの依存構造解析器では正解できなかったが, MWE を考慮した依存構造解析器では正解した事例を紹介する. **図 11**, **図 12** において, “officials” の head は正しくは “questions” であるが, オリジナルの依存構造解析器は MWE(“at least”) の構成トークン (“least”) を head として推定している. 一方, MWE を考慮した依存構造解析器は正解の “questions” を

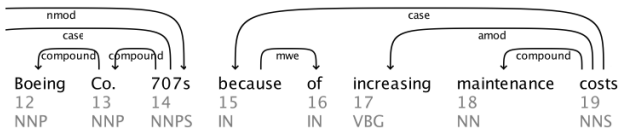


**図 13** オリジナルの依存構造解析器では正解していたが, MWE (“because of”) を考慮した依存構造解析器では正解できなかった事例：(a) オリジナルの解析器により推定された依存構造. “because” の head として, 正解の “costs” を推定している. 全文は “Earlier the company announced it would sell its aging fleet of Boeing Co. 707s because of increasing maintenance costs.”

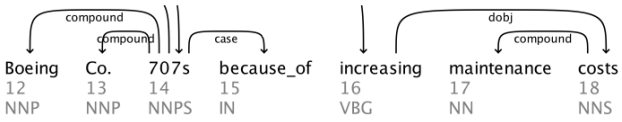


**図 14** オリジナルの依存構造解析器では正解していたが, MWE (“because of”) を考慮した依存構造解析器では正解できなかった事例：(b) MWE を考慮した解析器により推定された依存構造. “because of” の head として, MWE 直前の “707s” を推定している.

head として推定している. この事例では MWE を事前に認識した上で依存構造解析を行う事によって, MWE 構成トークンの品詞タグに起因する, 誤った係り受けの推定を回避出来ており, これは Nivre ら [8] がスウェーデン語の複合機能語付近の係り受けに関するエラー解析で述べている内容と整合している.

次に, オリジナルの依存構造解析器では正解していたが, MWE を考慮した依存構造解析器では正解できなかった事例を紹介する. **図 13**, **図 14** において, オリジナルの依存構造解析器は “because” の head として正解の “costs” を推定しているが, MWE を考慮した依存構造解析器は “because of” の head として, MWE 直前の “707s” を推定している. また, “increasing maintenance costs” の syntactic head が前者では “costs” だが, 後者では “increasing” になっており, MWE の head の推定は MWE 周辺の係り受けの推定と相互に関連している事が分かる.

なお, テストセットには 289 回 MWE が出現しているが, この内 テストセットにしか出現しない MWE は 3 個であり, ほとんどの MWE は訓練データセットにも出現していた. 上記の “because of” についても訓練データセットに出現しており, 上記事例を正解する為には, MWE 専用の素性 (MWE 構成トークンの表層形, 品詞など) や 2 次以上の素性を依存構造解析器に入れる必要があると考える.

## 5. おわりに

本稿では複合機能語を考慮した英語の依存構造コーパスを作成し, 構築したコーパス [*3] を用いた依存構造解析を行った. 今後は MWE を考慮した依存構造解析の為の素性設計, MWE 認識と依存構造解析の統合, MWE を考慮した依存構造の利用が有効と見込まれる言語解析に取り組んで行きたい.

### 参考文献

[1] Sag, I. A., Baldwin, T., Bond, F., Copestake, A. and Flickinger, D.: Multiword Expressions: A Pain in the Neck for NLP, *In Proc. of the 3rd International Conference on Intelligent Text Processing and Computational Linguistics (CICLing)*, pp. 1–15 (2002).

[2] Abeill ´ e, Anne, Cl ´ ement, L. and Toussenel, F.: Building a Treebank for French, *Speech and Language Technology, Springer.*, Vol. 20, p. 165188 (2003).

[3] Green, S., Marneffe, M.-C. D., Bauer, J. and Manning, C. D.: Multiword Expression Identification with Tree Substitution Grammars : A Parsing tour de force with French, *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pp. 725–735 (2011).

[4] Candito, M. and Constant, M.: Strategies for Contiguous Multiword Expression Analysis and Dependency Parsing, *Association for Computational Linguistics.*, pp. 743–753 (2014).

[5] Shigeto, Y., Azuma, A., Hisamoto, S., Kondo, S., Kouse, T., Sakaguchi, K., Yoshimoto, A., Yung, F. and Matsumoto, Y.: Construction of English MWE Dictionary and its Application to POS Tagging, *Proceedings of the 9th Workshop on Multiword Expressions*, No. June, pp. 139–144 (2013).

[6] Schneider, N. and Al., E.: Comprehensive annotation of multiword expressions in a social web corpus, *Proc. of LREC.* (2014).

[7] Mcdonald, R., Nivre, J., Quirmbach-brundage, Y., Goldberg, Y., Das, D., Ganchev, K., Hall, K., Petrov, S., Zhang, H., Täckström, O., Bedini, C., Castelló, N. B. and Lee, J.: Universal Dependency Annotation for Multilingual Parsing, *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pp. 92–97 (2013).

[8] Nivre, J. and Nilsson, J.: Multiword units in syntactic parsing, *Workshop on Methodologies and Evaluation of Multiword Units in Real-World Applications*, pp. 39–46 (2004).

[9] Marneffe, M.-c. D. and Manning, C. D.: Stanford typed dependencies manual, *20090110 Httpnlp Stanford*, Vol. 40, No. September, pp. 1–22 (online), available from ⟨http://nlp.stanford.edu/downloads/dependencies_manual.pdf⟩ (2010).

[10] McDonald, R., Pereira, F., Ribarov, K. and Hajič, J.: Non-projective dependency parsing using spanning tree algorithms, *HLT '05: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, Vol. 18, No. October, pp. 523–530 (2005).

[*3] 今回構築したコーパスは [11] で公開する予定である.

[11] Kato, A., Shindo, H. and Matsumoto, Y.: MWE Aware Dependency, (online), available from ⟨https://github.com/naist-cl-parsing/mwe-aware-dependency⟩.