NAIST-IS-DD1661004

Doctoral Dissertation

Construction and Analysis of Multiword Expression-Aware Dependency Corpus

Akihiko Kato

December 13, 2019

Graduate School of Information Science
Nara Institute of Science and Technology

A Doctoral Dissertation
submitted to Graduate School of Information Science,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
Doctor of ENGINEERING

Akihiko Kato

Thesis Committee:

Professor Yuji Matsumoto	(Supervisor)
Professor Satoshi Nakamura	(Co-supervisor)
Associate Professor Masashi Shimbo	(Co-supervisor)
Assistant Professor Hiroyuki Shindo	(Co-supervisor)

Construction and Analysis of Multiword Expression-Aware Dependency Corpus*

Akihiko Kato

Abstract

Multiword expressions (MWEs) consist of multiple words with syntactic or semantic non-compositionality. In downstream tasks exploiting syntactic dependency information and requiring the understanding of the meaning of the texts, MWE-aware dependency structures, where each MWE is a syntactic unit are preferable to word-based dependency structures. An English dependency corpus is often acquired with automatic conversion from a treebank of phrase structure trees. However, most of existing English treebanks do not guarantee that an MWE span corresponds to a phrase structure subtree. Hence, it is not straightforward to get MWE-aware dependency trees from these treebanks. To deal with this problem, I formalize procedures to ensure that an MWE span corresponds to a phrase structure subtree by modifying phrase structure trees, and I apply these procedures to the Ontonotes corpus to develop a dependency corpus in which an MWE is treated as a syntactic node. I focus on functional MWEs, adjective MWEs, and named entities.

The above MWEs always have continuous occurrences. However, in downstream tasks, it is also important to recognize verbal MWEs (VMWEs) such as phrasal verbs, which are likely to have discontinuous occurrences. Therefore, I conduct VMWE annotations on Ontonotes with crowdsourcing. Finally, I address the task to predict both continuous MWE-aware dependency trees and VMWEs [1]. The reason I deal with these two sub-tasks simultaneously is that

*Doctoral Dissertation, Graduate School of Information Science, Nara Institute of Science and Technology, December 13, 2019.

[1]Here, I define continuous MWEs as MWEs that always have continuous occurrences.

I can expect dependency information to be used as effective features in VMWE recognition. I perform experiments with continuous MWE-aware dependency corpus and VMWE annotations on Ontonotes. Experimental results demonstrate the effectiveness of a model based on the hierarchical multi-task learning of the following three tasks: continuous MWE recognition, a prediction of a word-based dependency tree that encodes MWE spans, and VMWE recognition.

Keywords:

Linguistic Resource, Corpus Construction, Syntactic Parsing, Dependency Parsing, Multiword Expressions

複単語表現を考慮した依存構造コーパスの構築と解析*

加藤 明彦

内容梗概

複単語表現（Multiword Expression, MWE）は，統語的または意味的な非構成性を有する複数の単語からなるまとまりである．統語的な依存構造の情報を利用し，かつ意味理解が必要なタスクでは，単語ベースの依存構造よりも，MWE を考慮した依存構造，即ち MWE を統語的な単位とする依存構造の方が好ましい．英語の依存構造コーパスは句構造コーパスからの自動変換によって構築される事が多いが，ほとんどの句構造コーパスでは，MWE が句構造の部分木になっていることは保証されていないため，MWE を考慮した依存構造を容易に得ることはできない．そこで本研究では，MWE が句構造の部分木になるように木を修正する手続きを定式化し，複合機能語と形容詞 MWE または固有表現を考慮した依存構造コーパスを Ontonotes 上に構築した．また意味理解が必要なタスクでは，複合機能語や形容詞 MWE のように，連続な出現のみを持つ MWE（連続 MWE）だけでなく，句動詞などの非連続な出現を持ちうる MWE（動詞 MWE）の認識も重要であるため，クラウドソーシングを用いて Ontonotes コーパスに対して動詞 MWE の注釈を行なった．最後に上記コーパスを利用し，連続 MWE を考慮した依存構造と動詞 MWE の双方を予測する問題に取り組んだ．これは，依存構造の情報は動詞 MWE 認識で有効な特徴量として働くという直感に基づいている．評価実験の結果，連続 MWE 認識，連続 MWE の範囲を依存関係ラベルとして符号化した依存構造解析，動詞 MWE 認識の階層的マルチタスク学習に基づくモデルの有効性が確認された．この結果は，連続 MWE を考慮した依存構造解析器が捉えた特徴量が，動詞 MWE 認識で有効である事を示唆している．

キーワード

言語資源, コーパス構築, 構文解析, 依存構造解析, 複単語表現

*奈良先端科学技術大学院大学 情報科学研究科 博士論文, 2019 年 12 月 13 日.

Contents

List of Figures

List of Tables

Acknowledgement

はじめに，指導教官である松本裕治先生に感謝致します．振り返ってみると，研究の大きな方向性を決める際，常に松本先生から的確なアドバイスを頂いていたように思います．学生が自主独立して，自由にかつ楽しんで研究を行うことを先生は重視されており，その気風が研究室全体に良い空気感をもたらしていると感じております．

次に，准教授の新保仁先生に感謝致します．新保先生が主催されている DMLA 勉強会では，主に機械学習的な観点での新規性を持つ論文が紹介される事が多く，勉強会への参加を通して視野が広がり，論文を批判的に読むことの重要性を学ぶことができたと感じています．また，短期間ではありましたが，日本語データを扱うプロジェクトにも参加させて頂きました．現在の自然言語処理分野の研究では，どうしても英語のデータセットを扱うことが多く，日本語で NLP タスクを解くためのデータセット構築については，まだまだ課題が残されていることを実感できたという意味でも貴重な経験でした．

次に，助教の進藤裕之先生に感謝致します．進藤先生には定期的にミーティングを行って頂き，研究の進め方や論文執筆の細かい点を含め，丁寧なご指導を頂きました．また，研究がなかなか進まず，モチベーションが上がらない時でも，こういう面白いトピックがあるよ，といったお声がけを頂き，救われた事も多かったです．初めて参加した国際会議 (LREC (2016)) では，色々，話をして頂き，ありがたく感じております．

また，秘書の北川祐子さんに感謝致します．北川さんには，学会参加や RA の報告書などの事務手続きで大変お世話になりました．その手堅い仕事の進め方に，学ぶことが多かったです．

次に，研究室の先輩方に感謝致します．大内さんからは，その活発な研究姿勢や，コンスタントに研究実績を挙げられている所と，後輩への面倒見の良さを拝見させて頂き，大きなエネルギーを頂きました．また，近藤さん，椿さん，重藤さん，濱口さんは，DMLA 勉強会や研究会で，興味深い研究発表や論文紹介を行っておられ，発表を聴くのが楽しみでした．特に修士の時，先輩に勉強会などで教えて頂く機会が多く，とても助かりました．吉本さん，大村さん，小林さんとは CICP でゲーム AI の開発を通じて，楽しい経験をさせて頂きました．吉本さんからは，特に開発上のスキルなどを学ばせて頂きました．どうもありがとうございます．

また，同期の澤井くんからは，鋭いコメントを折に触れてもらい，競技プログラミング勉強会などでも議論させてもらいました．M1 の時に，彼と色々話す事ができて，そこから自分の研究生活が始まっていったと考えています．どうもありがとう．駒井くんからは，MWE 勉強会で有用なコメントを頂きました．自分の研究でも彼の構築したリソースを利用させてもらっていて，感謝しています．三田くんは，高いモチベーションで，誤り訂正の研究に取り組んでいて，そのパフォーマンスとアクティブな姿勢に学ばされる事が多かったです．Phi Van-Thuy さんとは，M1 の授業が一緒だったり，新保先生のプロジェクトで協力したりと色々，交流させてもらいました．少し時期はずれますが，同一年度に共に修了する事ができ，とても励みになっています．博士後期課程から同期となった，Ander さん，An さん，森元さん，劉さん，Michael さんとも折りに触れ，Discussion や雑談をさせてもらいました．忘年会後の二次会は Emotional なものでした．同期の皆の，今後の活躍を心から祈念しています．

後輩にも大変優秀な方々が多く，日々の研究生活を通して色々，学ばせてもらっています．吉川くんとは競技プログラミング勉強会でご一緒させてもらい，寺西くんとは，MWE や並列句，依存構造解析について議論させてもらいました．井上くんとは，言語横断的な情報抽出についての CICP プロジェクトを行い，LREC (2018) でもご一緒しました．追いコン後に石原くんも交えラーメンを食べに行ったり，井上くんの送別会後に，本多くんと三人で映画を 3 本観たのも良い思い出です．Tran さん，本多くん，和田くん，柴原くんとも随時，Discussion させてもらっています．興味のあるトピックがそれぞれ異なっているので，話していて面白いです．

また，大内さん，吉川くん，井上くん，佐藤くんと ACL (2017) に参加できたのは，とても良い経験になりました．日本に留まらず国際的に，未来の NLP に貢献するであろう人々と交流できた事は自分にとって大きな財産だと考えています．ここに感謝致します．

奈良で 5 年半過ごす中で，研究室の外でも，多くの方に助けて頂きました．奈良先端大の David Sell 先生には，国際会議原稿の英文校正を行って頂きました．ここに感謝致します．また，両親の友人が生駒市に住んでおられ，折に触れて，食事会などに誘って頂きました．深く感謝致します．

また，科学・工学の道に入るきっかけを作ってくれた，祖父・久雄に感謝します．その自由な生き方は，今に至るまで，自分の根本的なルーツとなっています．本当にありがとう．また，残念ながら鬼籍に入られた，私を教え導いてくれ

た数多くの方々に，深く感謝します．研究も含め，これからの自分の活動によって，何がしかをこの世界に貢献する事を通して，恩返しできればと考えています．

最後になりましたが，両親と，いつもそばで支えてくれている妻・香菜子に感謝致します．三年前に結婚を機に関東から来てくれ，色々，不便な事もあったかもしれないけど，少しずつ生活が軌道に乗ってきたかなと感じています．今までありがとう．そしてこれからもよろしくね．

1 Introduction

Multiword expressions (MWEs) consist of multiple words with syntactic or semantic non-compositionality. MWEs could be classified into the following four categories based on their grammatical roles (Table 1): (1) functional MWEs [2] (*a number of*, *even though*), (2) adjective MWEs (*dead on one' s feet*, *out of business*), (3) verbal MWEs (*pick up*, *make a decision*), and (4) compound nouns (*traffic light*). Hereafter, a grammatical role of an MWE is referred to as an MWE-level part-of-speech tag (MWE-POS tag).

In downstream tasks exploiting syntactic dependency information and requiring the understanding of the meaning of the texts, MWE-aware dependency structures (Figure 1b) are preferable to word-based dependency structures (Figure 1a). While each MWE is a syntactic unit in MWE-aware dependency structures, word-based dependency structures do not represent MWE spans.

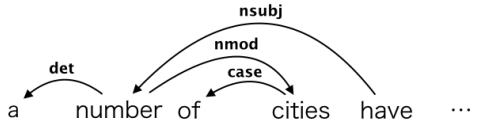
As an example of downstream tasks that could enjoy the advantages of MWE-aware dependency structures, I can mention event extraction, which involves event trigger detection and event arguments identification (Björne et al., 2017). Both of event triggers and arguments could be MWEs. Besides, the shortest dependency path connecting a trigger and an argument is often used as features of argument identification systems (Li et al., 2013).

Traditionally, an English dependency corpus has been acquired with automatic conversion from a treebank of phrase structure trees, such as Penn Tree-

Categories	**Examples**
Functional MWEs	a number of, even though, after all
Adjective MWEs	dead on one's feet, out of business
Verbal MWEs	pick up, look for, make a decision
Compound nouns	customer service, traffic light

Table 1: Categories of MWEs.

[2] I define functional MWEs as MWEs that function either as prepositions, conjunctions, determiners, pronouns, adverbs, auxiliary verbs, to-infinitives, or interjections.



(a) A word-based dependency structure

(b) An MWE-aware dependency structure

Figure 1: Word-based and MWE-aware dependency structures. The former does not represent an MWE span ("a number of"). In the latter, the MWE is represented as a single node.

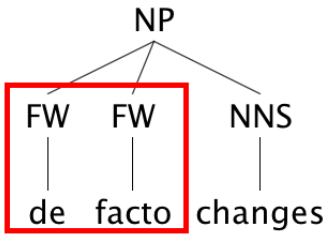


Figure 2: An example in which an MWE span does not correspond to a phrase structure subtree.

bank (Marcus et al., 1994). However, most of existing English treebanks do not guarantee that an MWE span corresponds to a phrase structure subtree (Figure 2). Hence, one could not acquire MWE-aware dependency trees by simply merging component words of each MWE in dependency trees converted from phrase structure trees.

To deal with this problem, I formalize procedures to ensure that an MWE span corresponds to a phrase structure subtree by modifying phrase structure trees (Chapter 3), and I develop a dependency corpus that is aware of functional MWEs (Chapter 3) and either adjective MWEs (Chapter 6) or named entities (Chapter 4) in Ontonotes 5.0 (Pradhan et al., 2007).

MWE-aware dependency parsing is different from word-based dependency parsing in that the former includes MWE recognition because an MWE-aware dependency tree treats an MWE as a syntactic unit. To explore models that are suitable for MWE-aware dependency parsing, I compare performances of the following two models (Chapter 4): (1) a pipeline model of MWE recognition with

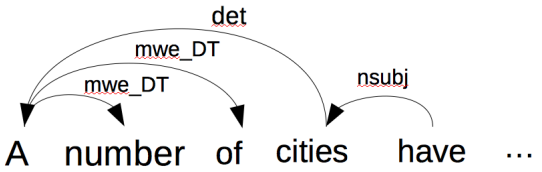


Figure 3: An example of a head-initial dependency tree, in which a span of an MWE ("a number of") and an MWE-level POS-tag ("DT") are encoded as dependency labels ("mwe_DT").

the sequential labeler and MWE-aware dependency parsing, and (2) a word-based dependency parsing to predict a tree that encodes MWE spans as dependency labels (head-initial dependency parsing)(Figure 3). I evaluate the above two models by using a dependency corpus that is aware of named entities (NEs) and functional MWEs [3] as a dataset. Experimental results show that the head-initial dependency parser improves MWE recognition compared to the pipeline model, with the help of features captured by the sequential labeler for MWE recognition.

Functional MWEs, adjective MWEs, and named entities always have continuous occurrences. However, in downstream tasks, it is important to recognize not only continuous MWEs [4] but also verbal MWEs (VMWEs) such as phrasal verbs, which are likely to have discontinuous occurrences (e.g., **take .. off**). Therefore, I conduct VMWE annotations on Ontonotes 5.0 with crowdsourcing (Chapter 5). To exploit crowdsourcing, I formalize VMWE annotations as a multiword-sense disambiguation problem.

Finally, I address the task to predict both continuous MWE-aware dependency trees and VMWEs (Figure 4) (Chapter 6). The reason I deal with these two subtasks simultaneously is that I can expect dependency information to be used as effective features in VMWE recognition. I perform experiments with a continuous MWE-aware dependency corpus and VMWE annotations on Ontonotes 5.0. Experimental results demonstrate the effectiveness of a model based on the hierarchical multi-task learning (HMTL) (Sanh et al., 2018) of the following three tasks: continuous MWE recognition, a prediction of head-initial dependency trees, and

[3]This corpus is available at: `https://catalog.ldc.upenn.edu/LDC2017T16`.

[4]Here, I define continuous MWEs as MWEs that always have continuous occurrences.

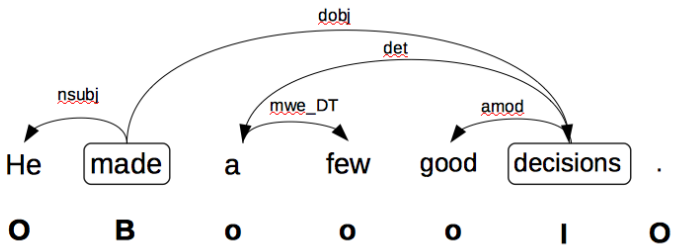


Figure 4: A sentence that includes both continuous ("a few") and discontinuous MWEs ("made .. decisions"). For this sentence, a model predicts both continuous MWE-aware dependency trees (shown in the upper half of this figure) and verbal MWEs (VMWEs). In the bottom half of this figure, I show a sequence of extended BIO tags (Schneider et al., 2014), which can represent gaps between components of a VMWE with "o" (small-o).

VMWE recognition.

2 Preliminaries

2.1 MWE recognition

2.1.1 Continuous MWE recognition

Continuous MWE recognition (CMWER) is a task to predict spans of CMWEs in a given sentence [5]. CMWER is typically formalized as a sequence labeling task by using BIO or BIOUL tagging schemes. Similar to other sequence labeling problems, recent models for CMWER adopt the bi-LSTM-CNNs-CRF model (Ma and Hovy, 2016).

2.1.2 Discontinuous MWE recognition

Discontinuous MWE recognition is a task to predict discontinuous MWE occurrences in a sentence. Each discontinuous MWE is represented as a set of token indices, which could have a gap. Hereafter, "a group" is used to mean a set of token indices that belong to the same discontinuous MWE. Because components of a discontinuous MWE could have a gap between them, the recognition of possibly discontinuous MWEs, such as verbal MWEs (VMWEs) is not able to be formalized with BIO or BIOUL tagging schemes. However, if a gap between components of a discontinuous MWE is not nested [6], one can encode an occurrence of the MWE as a sequence of extended BIO tags (Schneider et al., 2014). These tags can represent gaps between components of an MWE with "o" (small-o).

2.1.3 Evaluation Measure

An evaluation of CMWER is based on the exact span matching, which is measured with span-level F1-scores. In contrast, VMWER models are evaluated with group-level F1-scores. Here, "a group" means a set of token indices that belong to the same discontinuous MWE (2.1.2).

[5] Here, I define continuous MWEs (CMWEs) as MWEs that always have continuous occurrences.

[6] In other words, if a gap is filled by another MWE, this nested MWE itself does not contain a gap.

2.2 Dependency Structure

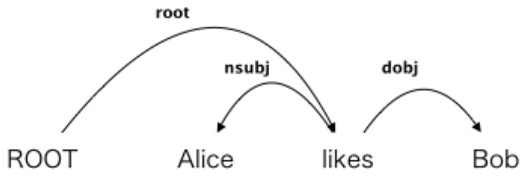
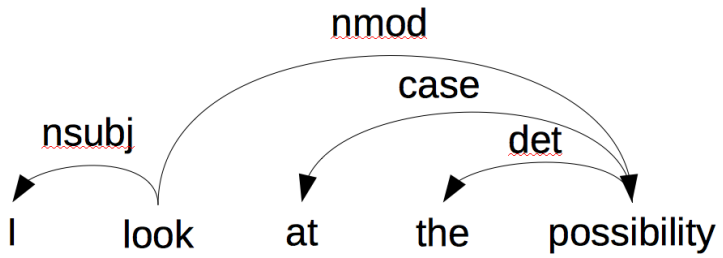


Figure 5: An example of a dependency structure.

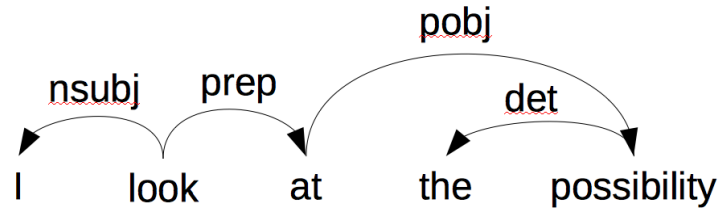
In the dependency grammar, a syntactic structure of a sentence is a set of binary, asymmetrical, and labeled relations between two lexical items (words), which is often called a head and a dependent because a head syntactically governs a dependent (Kubler et al., 2009). These relations are often called dependencies. A grammatical relation (e.g., subject and object) between the head and dependent is represented by a dependency label, which is also called a dependency relation. For instance, in Figure 5, "Alice" depends on "likes". Therefore, a dependency relation "nsubj" holds between the head ("likes") and the dependent ("Alice"). Because each dependent generally has a unique head, the set of dependencies in a sentence constitute a tree, often called a dependency tree. As shown in Figure 5, the artificial "ROOT" is added before the first word of a sentence. In this example, the dependent of "ROOT" is "likes", which is a real syntactic root of this dependency tree. This addition of "ROOT" leads to make every word have a syntactic head and to make it easy for a parser to handle each dependency tree.

2.2.1 Dependency schemes

A dependency treebank usually follows either content-head or function-head schemes. In the content-head scheme (Figure 6a), content words are directly connected with a dependency edge. Universal Dependencies (McDonald et al., 2013) and versions after ver.3.5.2 of Stanford Dependency (de Marneffe and Manning, 2008) adopt the content-head scheme. In contrast, in the function-head scheme (Figure 6b), a preposition works as an intermediate node between a modified word and an



(a) In the content-head scheme, content words ("look" and "possibility") are directly linked with each other.



(b) In the function-head scheme, a preposition ("at") works as an intermediate node between a modified word ("look") and an object of the preposition ("possibility").

Figure 6: Content-head and function-head schemes.

object of the preposition. Stanford basic dependency (de Marneffe and Manning, 2008) follows the function-head scheme.

2.2.2 Projectivity

A dependency arc (w_i, r, w_j) is projective if there is a directed path from w_i to every node between w_i and w_j. A dependency tree is projective if every edge is projective. Otherwise, the tree is non-projective. A projective dependency tree has no crossing edge.

2.3 Dependency Parsing

Dependency parsing is a task to predict a dependency tree for a given sentence. In a supervised setting, a dependency parser is trained on a set of pairs of a sentence

and a gold dependency tree. Dependency parsers are classified into graph-based and transition-based parsers.

2.3.1 Graph-based Parsing

In a graph-based parser, a dependency tree is factored into a set of subgraphs. In testing, the parser predicts a dependency tree based on a score calculated from scores of subgraphs. In most of the graph-based dependency parsing models, a score of a dependency tree is defined as a summation over scores of subgraphs.

The simplest graph-based parser is the arc-factored model. In this model, a subgraph described above is a single edge. The model parameters are defined for each edge (w_i, r, w_j). Here, w_i and w_j are tokens, and r is a dependency label.

An inference of the arc-factored model equals to find a maximum spanning tree (MST). The arc-factored model can be applied to both non-projective and projective dependency trees.

If I focus on models for graph-based non-projective dependency parsing, the most famous one is Chu-Liu-Edmonds Algorithm (McDonald et al., 2005). On the other hand, Eisner's algorithm (Eisner, 1996) is known as a classical method for a projective dependency parsing.

In an arc-factored model, one could use any features over each dependency edge. Standard features are followings: (1) word forms and part of speech (POS) tags of w_i and w_j, (2) the dependency label (r), (3) a distance between w_i and w_j, (4) a direction of the dependency edge, (5) POS tags of words near w_i or w_j (w_{i-1}, w_{i+1}, w_{j-1}, and w_{j+1}), (6) POS tags of words between w_i and w_j. For training of non-neural network parsers, an inference-based training, such as a perceptron algorithm (Collins and Roark, 2004) is often used.

Hereafter, I introduce recent neural graph-based dependency parsers.

2.3.1.1 Kiperwasser and Goldberg (2016)'s parser

Kiperwasser and Goldberg (2016) adopts an arc-factored model (McDonald et al., 2005), however, they replace a linear scoring function for each dependency arc in a candidate dependency tree with a neural network-based one. Concretely, for a given sentence s and a given potential arc (h,m), they firstly encode s with a bi-directional LSTM Hochreiter and Schmidhuber (1997) and get hidden vectors

for all tokens in s. After that, a score of (h,m) is calculated with an MLP which takes as an input a concatenation of a pair of hidden vectors corresponding to h and m.

2.3.1.2 Deep biaffine parser

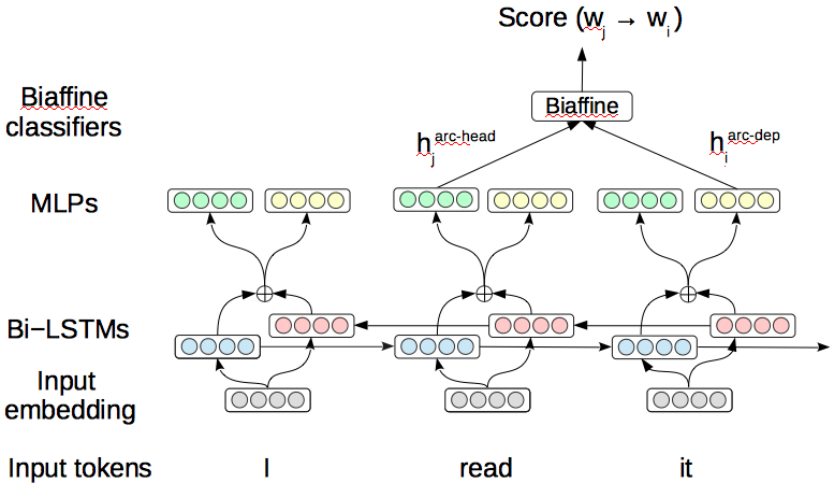


Figure 7: An architecture of the biaffine parser. With respect to MLPs, for brevity, I show only MLPs to calculate $h_i^{arc-dep}$ and $h_i^{arc-head}$.

By extending Kiperwasser and Goldberg (2016), Dozat and Manning (2017) proposes the deep biaffine parser, which consists of the following three components: a bidirectional LSTM (bi-LSTM), Multi-layer perceptrons (MLPs), and biaffine classifiers (Figure 7).

Given a sentence s, a bi-LSTM is used to get hidden states for all tokens in s, similar to Kiperwasser and Goldberg (2016).

Second, they produce four different vectors: $h_i^{arc-dep}$, $h_i^{arc-head}$, $h_i^{rel-dep}$, and $h_i^{rel-head}$ from each hidden state h_i using MLPs (Dozat et al., 2017).

Third, they predict a head of each token w_i in s by using a biaffine classifier, which takes $h_i^{arc-dep}$ and $h_j^{arc-head}$ as inputs. This classifier outputs a score for each potential dependency arc from w_j to w_i. The model predicts a head word $w_{j'}$ for a dependent w_i that have the highest score.

Finally, they predict a dependency label of a edge from a predicted head $w_{j'}$ to w_i with another biaffine classifier, which takes $h_i^{rel-dep}$ and $h_{j'}^{rel-head}$ as inputs. An output of this classifier is a score for each potential dependency label.

They perform joint training of the above two biaffine classifiers with an objective which is the sum of their softmax cross-entropy losses. The decoding could be done by finding a maximum spanning tree (MST) of tokens of a test sentence.

2.3.2 Transition-based Parsing

In a transition-based approach, dependency parsing is formalized as a sequence of state-transitions. Each state of a transition system, which is often called a configuration, is defined as the triple: $c = (\sigma, \beta, A)$ where σ is called "stack", β is called "buffer", and A is a set of dependency arcs. A stack and a buffer can contain tokens of a sentence. An initial state is $c_0 = (w_0, (w_1,...,w_n), \phi)$ (Figure 8a). Here, w_0 is an artificial ROOT, $(w_1,...,w_n)$ are tokens of a sentence. On the other hand, a terminal state is $c_f = (\sigma, \phi, A)$ (Figure 8d).

For each state, a transition system chooses one of the possible state-transitions, also called "actions". There are some variants for a set of actions. For brevity, I explain a basic shift-reduce transition system, which is known as Arc-Standard (Nivre, 2004). In this system, possible actions are Shift, Left-Arc, and Right-Arc. Shift moves the first token in a buffer to the top of the stack (Figure 8b). Left-Arc adds to A a dependency edge from b_0 (the first token in a buffer) to s_0 (the top of the stack), and remove s_0 from the stack (Figure 8c). Right-Arc adds to A a dependency edge from s_0 to b_0, replaces b_0 with s_0, and remove s_0 from the stack.

In many parsing models, each configuration is mapped to a feature vector. Standard features are word forms, lemmas, POS tags, dependency labels of various positions in a transition system, that is, each position of a stack, a buffer and leftmost/rightmost children of s_0 and b_0.

Based on a feature representation and model parameters, a parser predicts the next action. In training, a parser optimizes model parameters based on the training set: a set of pairs of a sentence and a gold state-transition sequence. In testing, a parser predicts a state-transition sequence based on the trained parsing model.

In recent years, many neural transition-based dependency parsers are pro-

posed. Chen and Manning (2014)'s work is the first attempt to use a neural network for dependency parsing. They use a standard neural network with one hidden layer to predict the next action for a given configuration. Input features are a concatenation of word, POS, and dependency label embeddings from a set of elements based on the stack and buffer positions for each type of information (word, POS or dependency label). Second, Andor et al. (2016) proposes a globally normalized transition-based neural network model with a beam search and a conditional random field (CRF) objective (Lafferty et al., 2001). Even though they use a simple feed-forward neural network, performances of their model are comparable or better than LSTMs because their network is globally normalized. Finally, Ma et al. (2018) introduces stack-pointer networks, which consist of the following two steps. First, their proposed model reads and encodes the whole sentence with a bi-directional recurrent neural network (RNN) encoder. After that, it builds the dependency tree in a top-down (from root-to-leaf) and depth-first manner with a uni-directional RNN decoder with an internal stack. At each time step t, the decoder chooses a specific position in an input sentence according to attention scores that are calculated from encoder hidden states and a decoder hidden state at the time step t. After that, the decoder generates an arc between the head at the top of the internal stack and the selected token (dependent). Their proposed model is more efficient than graph-based parsers because it is a transition-based system, and it can have a global view of the whole sentence.

2.3.3 Evaluation Measure

Standard evaluation measures used in dependency parsing are unlabeled attachment score (UAS) and labeled attachment score (LAS). UAS represents how accurately the parser predicts a head for each token. In calculating UAS, whether the predicted dependency label is right or not is not taken into account. In contrast, LAS regards a predicted dependency edge for a given dependent to be correct only if both the head and dependency label equal to those of the gold standard.

2.4 Syntactic Corpora

2.4.1 Penn Treebank

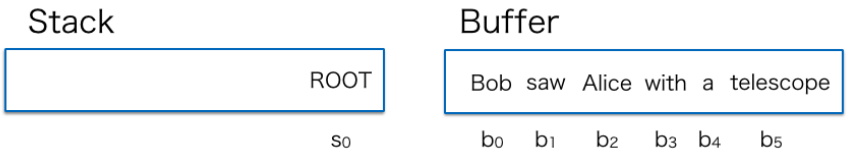
In the research of syntactic parsing, a manually annotated treebank is important as training, tuning, and testing data used by a syntactic parser. Penn Treebank (Marcus et al., 1994) is one of the de facto standard English treebanks. Sentences in this corpus originate from the Wall Street Journal (WSJ), Brown corpus, Switchboard, and ATIS.

The POS tagset of Penn Treebank is more simplified than that of Brown Corpus as some redundant tags have been merged. In Brown Corpus, for instance, "did" is assigned a special POS tag: DOD. On the other hand, a POS tag of "did" in Penn Treebank is the same as the past tense of other verbs: VBD.

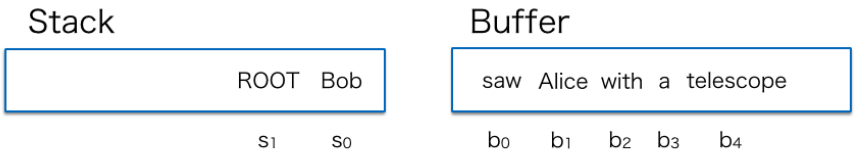
In addition to the standard syntactic tagset (e.g., NP and VP), Penn Treebank provides three kinds of functional tags: text categories (e.g., titles), grammatical functions (e.g., logical subjects in passives), and semantic roles (e.g., temporal phrase).

2.4.2 Ontonotes

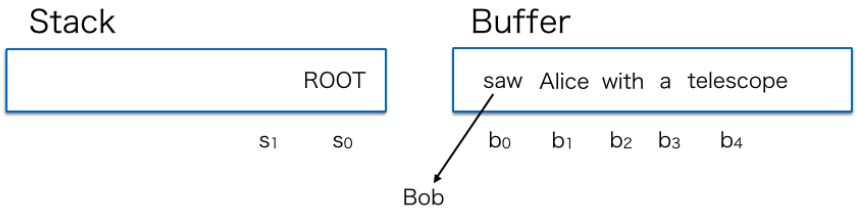
Ontonotes (Pradhan et al., 2007) is a corpus that provides multiple levels of annotations: syntax, propositions, word senses, named entities, and coreference. Ontonotes contains newswire data of English and Chinese. In particular, the English portion is a non-financial portion of WSJ. A syntactic layer is built on Penn Treebank.



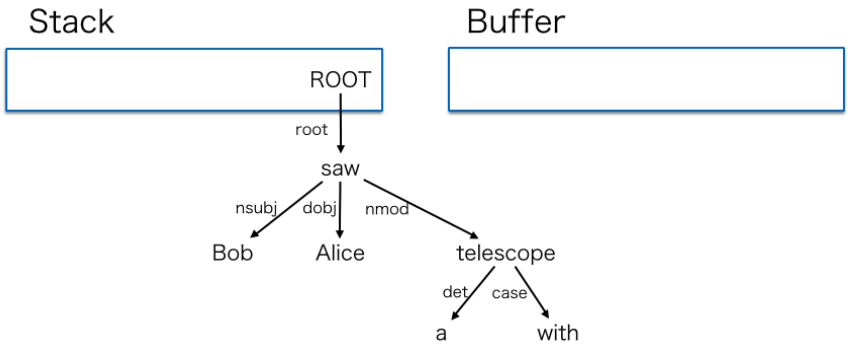
(a) An initial configuration of shift-reduce parser.



(b) A configuration after shift action by which "Bob" moved from b_0 to s_0.



(c) A configuration after Left-Arc action by which "Bob" was removed from s_0 and governed by "saw" at b_0.



(d) A terminal configuration. Full dependency tree is constructed.

Figure 8: A sequence of configurations of the transition-based dependency parser for a sample sentence ("Bob saw Alice with a telescope").

3 Construction and Analysis of Functional Multiword Expression-Aware Dependency Corpus

3.1 Introduction

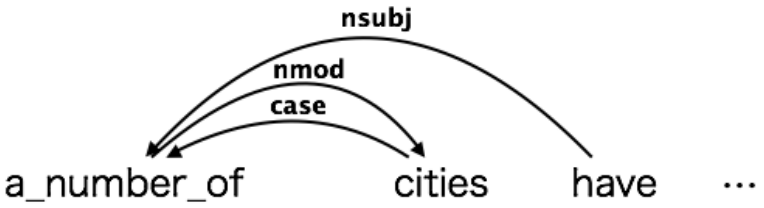


Figure 9: A cycle and multi-heads occur if one could combine nodes in the MWE ("a number of") into a single node.

As I describe in Chapter 1, given a phrase structure tree in which an MWE span does not correspond to a subtree, if one converts this phrase structure tree to a word-based dependency tree, it is not always possible to acquire an MWE-aware dependency tree by directly combining nodes belonging to an MWE into a single node in the word-based dependency tree.

The above method often leads to the following problems: a node derived from an MWE could have multiple heads and the whole dependency structure including an MWE might be cyclic [7](Figure 1a → Figure 9). This is mainly because Penn Treebank style annotation does not give special treatment for MWEs. I discuss this problem further in Chapter 3.3.

To tackle the above problems, in this chapter, I propose a three-step conversion method to get MWE-aware dependency structures. First, I establish an MWE as a subtree in a phrase structure (Figure 11a → Figure 11b). Second, I replace the subtree corresponding to the MWE by a preterminal with its leaf node as a child (Figure 12). The preterminal has an MWE-level POS tag. Its child node is made by joining all components of the MWE with underscores. As a final step, I convert the phrase structure to Stanford Dependency (de Marneffe and

[7]This problem often happens in the content-head scheme, such as Universal Dependencies (McDonald et al., 2013).

Manning, 2008). In this way, I can avoid an occurrence of multiple heads and/or cycles in an MWE-aware dependency tree, because an MWE constitutes a single node in this dependency tree. I apply this conversion method to the Ontonotes corpus (Pradhan et al., 2007) to construct a dependency corpus that takes MWEs into consideration. In this work, I focus on functional MWEs, which are one type of MWE that serves as functional expressions. This is because functional MWEs have a variety of functionalities that may affect language analyses such as parsing and POS tagging. To reduce the cost of annotation in constituting each MWE as a subtree, I classify patterns of MWEs as seen in phrase structures in terms of ease of conversion. I manually annotate only instances that are difficult to convert automatically.

Furthermore, I evaluate the performance of the first order MST Parser (McDonald et al., 2005) on the constructed MWE-aware dependency corpus. I get an unlabeled attachment score (UAS) comparable to that obtained on the original Ontonotes corpus. Moreover, I qualitatively analyze some test instances in which MWE-aware and original dependency parsers predict different dependency heads of an MWE (Chapter 3.4).

3.2 Related Work

Shigeto et al. (2013) creates a dictionary of English functional MWEs [8], and annotates functional MWEs that appear in Penn Treebank. In their corpus, each MWE occurrence has its MWE-level POS tag and a span of component tokens.

As an example of an MWE-aware dependency corpus, I introduce Universal Dependency Treebank (McDonald et al., 2013). This project is developing a cross-linguistically consistent treebank annotation for many languages. A functional MWE is annotated in a flat and head-initial structure, in which second and later words in an MWE modify the first word using a "fixed" label.

In MWE-aware syntactic parsing, an MWE is recognized before or at the

[8]Here, I refer to MWEs that function either as prepositions, conjunctions, determiners, pronouns, adverbs, auxiliary verbs, to-infinitives, or interjections as functional MWEs. Because the purpose of Shigeto et al. (2013) is to annotate fixed MWEs, their annotation does not include auxiliary verbal MWEs (aux-VMWEs) (e.g., "might have been"). I treat annotations and the integration of aux-VMWEs into dependency structures as future work.

time of syntactic parsing (Green et al., 2011; Candito and Constant, 2014). Nivre and Nilsson (2004) conducts Swedish dependency parsing, in which they assume perfect MWE recognition. They focus on multiword names (persons and places), numerical expressions and functional MWEs.

3.3 Construction of Functional MWE-aware Dependency Corpus

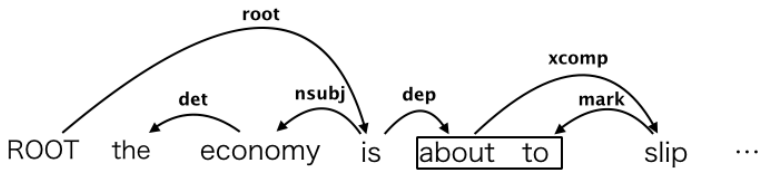


Figure 10: A cycle and multi-heads occur if one could combine nodes in the MWE (“about to”) into a single node.

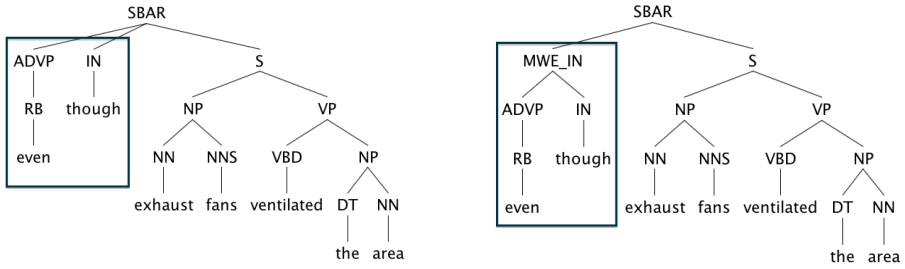
I build an MWE-aware dependency corpus based on the corpus by Shigeto et al. (2013). In their corpus, an entire MWE is assigned an MWE-level POS tag. In such a case, it is preferable that an MWE is a single node in a dependency structure. On the other hand, each word is treated as a node in a word-based dependency structure.

To directly convert word-based dependency to MWE-aware dependency, one needs to combine nodes in an MWE into a single node. However, this naive approach often leads to the following problems: a node derived from an MWE could have multiple heads and the whole dependency structure including an MWE might be cyclic. For instance, in Figure 9, “a_number_of” has multiple heads (“cities” and “have”), and a cycle occurs because of the following edges: “a_number_of” $\rightarrow$ “cities” and “cities” $\rightarrow$ “a_number_of”. Therefore, one needs to remove one of the edges to get a dependency tree. Another problem is the following: the syntactic head of “a number of cities” is “cities” in an MWE-aware dependency structure (Figure 1b), because “a_number_of” is a determiner. Nevertheless, one cannot get the correct dependent of “have” (i.e., “cities”) by the above naive method. Similarly, if one combines words of the MWE into a single

node in Figure 10, "about to" has multiple heads ("is" and "slip"), and a cycle occurs between "about to" and "slip".

To solve these problems, I adopt the following approach: first, I establish an MWE in a phrase structure tree as a subtree. After that, I convert a phrase structure to a dependency structure. With this method, I can avoid cycles and/or multiple heads.

Concretely, I build a functional MWE-aware dependency corpus according to the following method:



(a) An LCA-tree before conversion. The square in this figure indicates the span of the MWE.

(b) The LCA-tree after conversion. Here, the MWE ("even though") becomes a subtree.

Figure 11: Conversion of an LCA-tree in a "Multiple contiguous children" case.

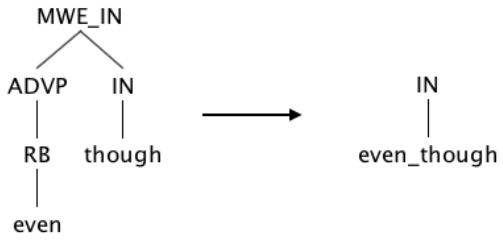


Figure 12: Replacement of a subtree in which the MWE ("even though") is grouped.

(1) Find an MWE in a phrase structure tree in Ontonotes and establish it as a subtree (Figure 11a → Figure 11b)[9].

[9]I utilize MWE annotations provided by Shigeto et al. (2013).

(2) Replace the above subtree by a preterminal with its leaf node as a child. The preterminal has an MWE-level POS tag. Its child node is made by joining all components of the MWE with underscores, as in Figure 12.

(3) Convert a phrase structure to Stanford Dependency (de Marneffe and Manning, 2008)[10].

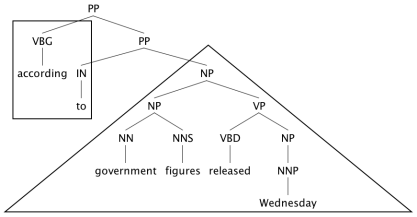
In Step (1), I convert an MWE into a subtree in the phrase structure tree. For example, "even though" in Figure 11a is annotated as an MWE in Shigeto et al. (2013). I convert it as in Figure 11b. If I can convert the span of an MWE into a subtree without changing other subtrees, I classify this instance as "Multiple contiguous children". Otherwise, the instance is treated as "Crossing brackets". When I group an MWE, I focus on the LCA-tree, which is the subtree rooted in the Least Common Ancestor (LCA) of the components of the MWE. In Figure 11a, the LCA-tree is rooted in SBAR, which is the LCA of "even" and "though".

The above method relates to Finkel and Manning (2009). For joint parsing and named entity recognition, they classify named entities that do not correspond to a phrase in the constituency tree to the following two categories. A named entity belonging to the first category is contiguous multiple children of some non-terminal. This category corresponds to the above "Multiple contiguous children" case. On the other hand, a span of each named entity belonging to the second category crosses brackets in the parse tree. It corresponds to the above "Crossing brackets" case.

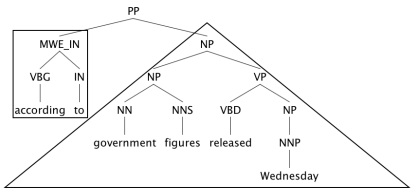
3.3.1 Multiple contiguous children

In the "Multiple contiguous children" case, I insert a new internal node under the LCA (Figure 11a → Figure 11b). This internal node covers precisely the span of the MWE.

[10]I designate "-conllx -basic -makeCopulaHead -keepPunct" as an option for the conversion command.



(a) An LCA-tree before conversion. The span of the right subtree (from "to" to "Wednesday") partially overlaps with the span of the MWE ("according to"). The triangle in this figure indicates the subtree (T_{post}) which covers precisely the span excluding the MWE.



(b) The LCA-tree after conversion. Here, the MWE ("according to") becomes a subtree. The triangle in the figure indicates the subtree (T_{post}).

Figure 13: Conversion of an LCA-tree in the "Regular" case.

3.3.2 Crossing brackets

In the "Crossing brackets" case, if I convert the MWE into a subtree (e.g. Figure 13a → Figure 13b), the structures of other subtrees need to be changed. For example, it is reasonable to remove the right child of LCA (PP) in Figure 13a at the time of the conversion as in Figure 13b.

Further, I classify "Crossing brackets" into "Regular" and "Irregular" cases based on the extent to which the LCA-tree is changed by the conversion.

Regular Case

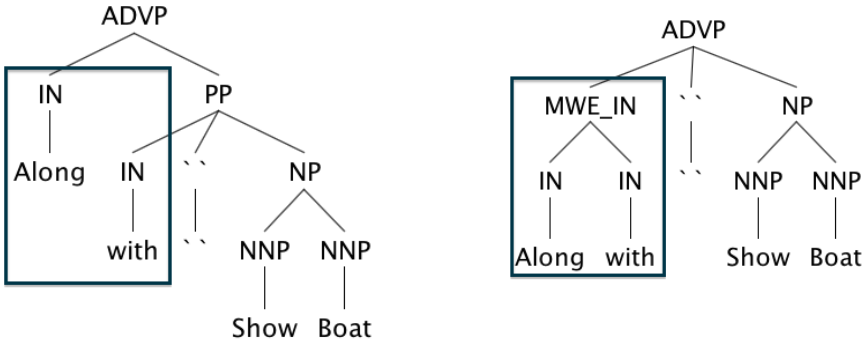
In this case, I can convert an MWE into a subtree without changing subtrees which cover words outside the MWE. In Figure 13a, the span of the right subtree (from "to" to "Wednesday") partially overlaps with the span of the MWE ("according to"), but there is an internal node which covers the entire span to the right of the MWE (see the grandchild of the LCA (NP)).

I define the subtree covering the span to the left of the MWE, the subtree covering the MWE, and the subtree covering the span to the right of the MWE as T_{pre}, T_{mwe}, and T_{post}, respectively. In Figure 13a, the subtree surrounded by a

triangle is T_{post}.

I convert the tree so that the LCA has T_{pre}, T_{mwe}, T_{post} as children (Figure 13a → Figure 13b).

When both T_{pre} and T_{post} are present, I manually determine which of the following choices is preferable: LCA having T_{pre}, T_{mwe}, T_{post} as flat children, or LCA having a new internal node (covering T_{mwe} and T_{pre} or T_{post}) as a child.



(a) An LCA-tree before conversion. There is no internal node which covers precisely the entire span to the right of the MWE ("along with").

(b) The LCA-tree after conversion. Here, the MWE ("along with") becomes a subtree.

Figure 14: Conversion of an LCA-tree in the "Irregular" case.

Irregular Case

In this case, I cannot avoid to change subtrees outside the MWE when converting the MWE into a subtree (Figure 14a → Figure 14b). The span of the right subtree of LCA (from "with" to "Boat") partially overlaps with the span of the MWE ("along with"), and there is no internal node that covers precisely a span to the left or right of the MWE. Hence, I manually determine how to combine a subtree covering the MWE with subtrees covering spans excluding the MWE.

Regarding "Multiple contiguous children" and "Crossing brackets (Regular)"

Case	No. of instances
Multiple contiguous children	1,663
Crossing brackets (Regular)	1,742
Crossing brackets (Irregular)	57

Table 2: Corpus statistics of an MWE-aware Dependency Corpus.

cases, I decide a symbol of an LCA as follows [11]:

$$\arg\max_{X_{LCA}}(\Pr(A \rightarrow B_1..X_{LCA}..B_m) \times \Pr(X_{LCA} \rightarrow C_1..C_n)) \quad (1)$$

With the method described above, I develop the MWE-aware dependency corpus based on the Wall Street Journal (WSJ) portion of Ontonotes 5.0. I show the corpus statistics in Table 2 [12].

3.4 MWE-aware Dependency Parsing

In this chapter, I perform dependency parsing by using either the original Ontonotes or the MWE-aware dependency corpus I build according to the method described in Chapter 3.3.

3.4.1 Experimental Setting

I trained and tested original and MWE-aware dependency parsers independently. The training and test data of the original dependency parser are from sections 02-21 and 23 of the original Ontonotes corpus, respectively. In contrast, the training and test data of the MWE-aware dependency parser are from sections 02-21 and 23 of the MWE-aware dependency corpus. In MWE-aware dependency parsing, I assume perfect MWE recognition and treat each MWE as a single node in both training and testing [13]. As the evaluation measure, I adopted the UAS (unlabeled attachment score). I used the first-order MST Parser (McDonald et al., 2005)

[11] These probabilities are calculated from Ontonotes.

[12] In Crossing brackets (Regular) case, 53 instances have both T_{pre} and T_{post}.

[13] I investigate models to predict both MWE spans and MWE-aware dependency trees in Chapter 4.

	UAS (total)	UAS (sentences including MWEs)
Original Ontonotes	89.99	87.77
MWE-aware dependency corpus	**90.01**	**87.84**

Table 3: Experimental results for the original and MWE-aware dependency parsing.

and the standard split for the WSJ portion of Ontonotes 5.0: sections 02-21 for training, 23 for testing. I also used gold POS tags both in training and testing.

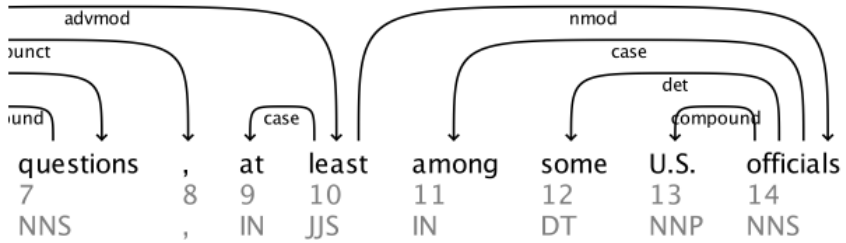
3.4.2 Experimental Results

I show the experimental results in Table 3. Because the original and MWE-aware dependency parsers use different test sets, I cannot directly compare the results of these dependency parsers. However, I got comparable UAS for the two parsers on both 1,640 sentences of the whole test set and 266 sentences including MWEs [14].

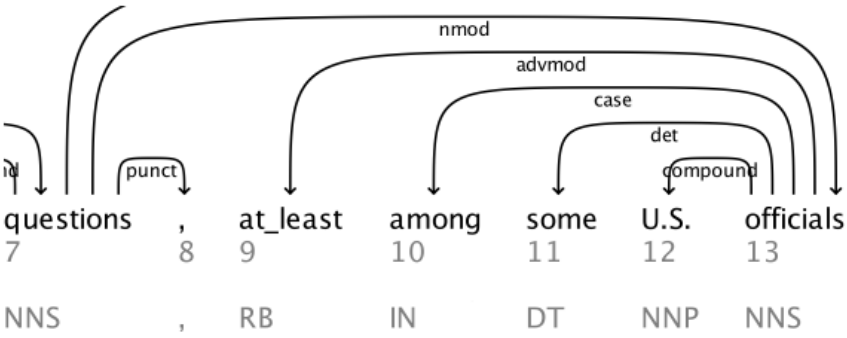
In the following, I describe the qualitative analysis. First, I show an instance for which the original and MWE-aware dependency parsers inferred incorrect and correct outputs, respectively. In Figure 15, the original dependency parser incorrectly infers "least", which is a component of the MWE ("at least"), as the head of "officials". On the other hand, the MWE-aware dependency parser correctly infers "questions" as the head of "officials" [15]. In this instance, by recognizing the MWE before parsing, the MWE-aware dependency parser achieves a correct prediction without influenced by POS tags of components of the MWE. This is consistent with the error analysis by Nivre and Nilsson (2004) for Swedish dependency parsing involving functional MWEs.

[14]Because the number of tokens belonging to MWEs corresponds to about 1.9 % of that of the whole test set, these two parsers could show similar performances for the whole test set, even though these models perform differently in terms of UAS regarding dependency edges which connect inside and outside of MWEs.

[15]The full sentence is "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."



(a) A dependency structure inferred by the original dependency parser.

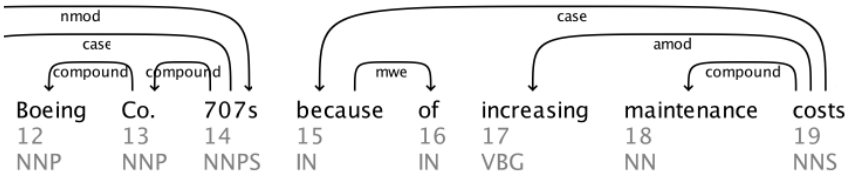


(b) A dependency structure inferred by the MWE-aware dependency parser.

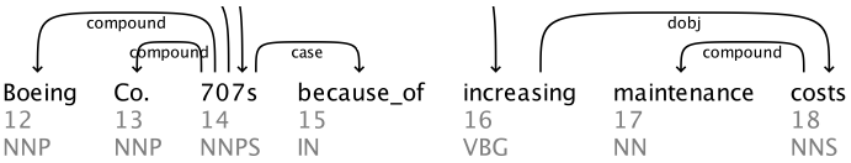
Figure 15: An instance for which the original dependency parser inferred an incorrect output and the MWE-aware dependency parser inferred a correct output.

Next, I show an instance for which the original and MWE-aware dependency parsers inferred correct and incorrect outputs, respectively. In Figure 16, the original dependency parser correctly infers "costs" as the head of "because". However, the MWE-aware dependency parser infers "707s" as the head of the MWE ("because of") [16]. This and almost all MWEs in the test set (286 out of 289 MWEs) also appear in the training set. Therefore, to perform a correct inference for the above instance, I need to explore MWE-specific features (the word form and POS tag of each component of the MWE) and/or higher-order features for the dependency parser rather than dealing with unseen MWEs.

[16]The full sentence is "Earlier the company announced it would sell its aging fleet of Boeing Co. 707s because of increasing maintenance costs."



(a) A dependency structure inferred by the original dependency parser.



(b) A dependency structure inferred by the MWE-aware dependency parser.

Figure 16: An instance for which the original dependency parser inferred a correct output, but the MWE-aware dependency parser inferred an incorrect output.

3.5 Summary

I created an English dependency corpus that is aware of functional MWEs and conducted dependency parsing using the constructed corpus [17]. As future work, I plan the followings: (1) I will design features for MWE-aware dependency parsing and address both of MWE recognition and dependency parsing, (2) I will explore a linguistic analysis in which an MWE-aware dependency tree is preferable to a word-based dependency tree, (3) Based on the fact that only 497 in 1,923 MWE types appear in the Ontonotes corpus, it is worthwhile to match the MWE dictionary with other large-scale corpus, such as Annotated English Gigaword (Napoles et al., 2012) and calculate what percentage of the MWE dictionary appears in the corpus at least once.

[17] This corpus is available at `https://catalog.ldc.upenn.edu/LDC2017T01`

4 Construction and Analysis of Dependency Corpus that is aware of Functional Multiword Expressions and Named Entities

4.1 Introduction

To solve complex natural language processing (NLP) tasks that require deep syntactic analysis, various levels of annotation such as parse trees and named entities (NEs) must be consistent with one another (Finkel and Manning, 2009). Otherwise, it is usually impossible to combine these pieces of information effectively.

In Chapter 3, to develop an MWE-aware dependency corpus, I establish the consistency between phrase structure trees and functional MWE spans. In other words, I ensure that a span of a functional MWE corresponds to a subtree of a phrase structure.

To pursue this direction further, in this chapter, I construct a corpus such that dependency structures are consistent with MWEs and NEs (MWE-Aware English Dependency Corpus 2.0 (MAED corpus ver.2.0) [18]), by extending Kato et al. (2016)'s corpus (MAED corpus ver.1.0 [19]), which I describe in Chapter 3. As is the case with MAED ver.1.0, each MME is a syntactic unit in an MWE-aware dependency structure in the MAED ver.2.0 (Figure 1b). Moreover, MAED ver.2.0 includes not only functional MWEs but also NEs. Because NEs are highly productive and occur more frequently than functional MWEs, they are difficult to cover in a dictionary.

Consistency between NE-spans and phrase structures is not guaranteed because they are independently annotated in most syntactic corpora. For instance, in Figure 17, a span of an NE, "Board of Investment" is inconsistent with the syntactic tree. Therefore, I resolve this inconsistency by modifying phrase structures locally and establishing each NE as a subtree.

Furthermore, to evaluate the constructed corpus, I explore pipeline and joint models that predict both MWE spans and an MWE-aware dependency tree [20].

[18] This corpus is available at: `https://catalog.ldc.upenn.edu/LDC2017T16`.

[19] `https://catalog.ldc.upenn.edu/LDC2017T01`

[20] In Chapter 3, I address MWE-aware dependency parsing based on gold MWE spans, which

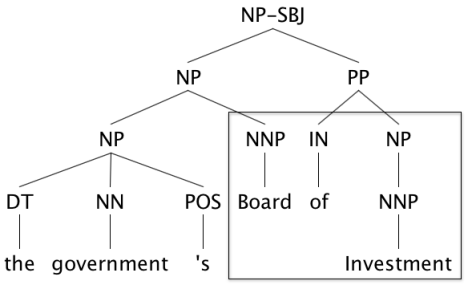


Figure 17: An example of inconsistency between NE-spans and phrase structures. A rectangle shows an NE-span.

MWE-level POS tags	NNP	RB	IN	others
MWE Instances	20,992	3,796	2,424	737
MWE Types	11,875	377	92	52

Table 4: Corpus statistics.

Experimental results show that the proposed joint model with additional MWE span features achieves an MWE recognition improvement of 1.35 points over the pipeline model.

Type of MWEs	Non-terminal	Multiple contiguous children	Crossing brackets
Functional MWEs	3,466	1,663	1,799
NEs	18,625	2,252	144

Table 5: Histogram with respect to the consistency between MWE spans and phrase structures.

is actually not a realistic scenario. By contrast, proposed models do not assume gold MWE spans.

4.2 MWE-aware Dependency Corpus

To ensure consistency between MWE annotations and dependency structures, I first integrate NE annotations in Ontonotes [21] into phrase structures such that functional MWEs are established as subtrees. Subsequently, I convert phrase structures to dependency structures. I construct my corpus by extending Kato et al. (2016)'s corpus (Chapter 3) [22], which is itself built on a corpus by Shigeto et al. (2013). Regarding MWE annotations, Shigeto et al. (2013) first constructs an MWE dictionary by extracting functional MWEs from the English Wiktionary [23], and classifies their occurrences in Ontonotes into either MWE or literal usage. Kato et al. (2016) integrates these MWE annotations into phrase structures and establishes functional MWEs as subtrees (Chapter 3).

Next, I describe the establishment of each NE as a subtree. If an NE-span does not correspond to any non-terminal in a phrase structure, there are two possibilities: (A) the NE-span corresponds to multiple contiguous children of a subtree, or (B) the NE-span has crossing brackets with the spans in the parse tree (Finkel and Manning, 2009; Kato et al., 2016). In Case (A), I insert a new non-terminal ("MWE_NNP") that governs the NE-span [24]. In Case (B), many instances correspond to a noun phrase (NP) comprised of a nested NP and a prepositional phrase (Figure 17). In the main NP, a modifier, such as a determiner, an adjective, or a possessive NP, precedes an NE. For these instances, according to Finkel and Manning (2009), I reduce Case (B) to Case (A) by moving the modifier from the nested NP to the main NP. Then, I establish each NE as a subtree by inserting an MWE-specific non-terminal ("MWE_NNP"). Furthermore, in some instances, it is more reasonable to enlarge NE-spans than to modify phrase structures. As a typical example, there is an NE annotation that covers only part of a coordination structure, such as "Peter and Edward Bronfman," where "Edward Bronfman" is annotated as an NE. In this case, I

[21] I exploit NE annotations in Ontonotes Release 5.0 (LDC2013T19). I address traditional NEs, such as persons, locations, and organizations, while omitting the following: DATE, TIME, PERCENT, MONEY, QUANTITY, ORDINAL, and CARDINAL. Note that I only focus on multiword NEs.

[22] `https://catalog.ldc.upenn.edu/LDC2017T01`

[23] `https://en.wiktionary.org`

[24] I do not require manual annotations for Case (A).

extend an original NE-span to the whole coordination structure. I show the statistics for the corpus in Table 9 [25]. This corpus has 27,949 MWE instances in 37,015 sentences. A histogram regarding the consistency between MWE spans and phrase structures is shown in Table 5. For tree-to-dependency conversion, I first replace a subtree corresponding to an MWE by a preterminal node and its child node. The preterminal node has an MWE-level POS tag (MWE_POS tags). The child node is generated by joining all components of the MWE with underscores. Then, I convert a phrase structure into a Stanford-style dependency structure (de Marneffe and Manning, 2008) (Figure 1b).

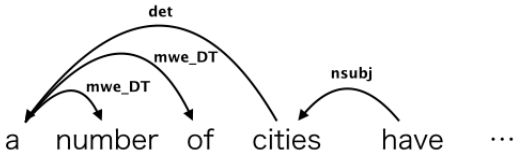


Figure 18: In the joint model, I directly infer an MWE-aware dependency tree in which an MWE ("a number of") is represented as a head-initial structure by a dependency parser.

4.3 Models for MWE identification and MWE-aware dependency parsing

In this chapter, I explore models that predict both MWE spans and an MWE-aware dependency structure (Figure 1b).

4.3.1 Pipeline Model

The pipeline model involves the following three steps. First, BIO tags encoding MWE spans and MWE_POS tags, such as "B_NNP" or "I_DT" are predicted by a sequential labeler based on conditional random fields (CRFs) (Lafferty et al., 2001). Second, tokens belonging to each predicted MWE span are concatenated into a single node. Finally, an MWE-based dependency structure (Figure 1b) is

[25] NEs have NNP as an MWE-level POS tag.

predicted by an arc-eager transition-based parser (Nivre, 2003). For the CRFs, in addition to word-form and character-based features, I use 1 to 3-gram features based on dictionaries of functional MWEs and NEs within 5-word windows from a target token. For a dictionary of functional MWEs, I use the dictionary by Shigeto et al. (2013) (Chapter 4.2). Meanwhile, I create a dictionary of NEs from a title list of English Wikipedia articles, except stop words, provided by UniNE [26]. Regarding parsing features, I use baseline features and rich non-local features proposed by Zhang and Nivre (2011).

4.3.2 Joint Model

In the proposed joint model, MWE spans and MWE_POS tags are encoded as dependency labels (Figure 18), and conventional word-based dependency parsing is performed by an arc-eager transition-based parser (Nivre, 2003). I use the same parsing features used in the pipeline model. I convert MWEs in MWE-aware dependency structures (Figure 1b) to head-initial structures (Figure 18) that encode MWE spans and MWE_POS tags. Note that this representation is similar to Universal Dependencies (McDonald et al., 2013). When parsing, I use constraints based on a history of transitions and the dictionary of functional MWEs. This is done to avoid invalid dependency trees. Because NEs are highly productive, I do not use a constraint regarding NEs.

Joint(+dict)

I design additional features based on matches with dictionaries of NEs and functional MWEs. Hereafter, I refer to the joint model coupled with these additional features as joint(+dict). For instance, given a sentence that starts with "a number of cities," the additional features are as follows: a / B_DT, number / I_DT, of / I_DT, cities / O. Based on these additional features, I extend the baseline features proposed by Zhang and Nivre (2011) to develop MWE-specific features whose atomic features include not only words and word-level POS tags, but also BIO tags encoding MWE spans and MWE_POS tags.

[26] `http://members.unine.ch/jacques.savoy/clef/englishST.txt`

	Dependency Parsing				MWE Recognition	
	All sentences		First tokens of MWEs			
Model	**UAS**	**LAS**	**UAS**	**LAS**	**FUM**	**FTM**
Pipeline	91.39	89.42	84.06	78.22	91.40	91.32
Joint	91.15	89.18	81.93	77.74	89.03	88.79
Joint(+dict)	91.36	89.37	84.45	80.74	91.93	91.78
Joint(+pred_span)	**91.50**	**89.51**	**84.85**	**81.29**	**92.75**	**92.60**

Table 6: Experimental results on the test set.

Joint(+pred_span)

Because dictionary matching is not concerned with context, in this setting, I use MWE spans and MWE_POS tags predicted by CRF, rather than dictionary matching. Hereafter, I refer to this as joint(+pred_span). By using features extracted from CRF predictions, I can mitigate error propagation from sequential labeling and consider information from a full sentence. Moreover, I can alleviate difficulties in predicting MWE spans and MWE_POS tags encoded as head-initial structures (Figure 18) by the parser.

4.4 Experimental Setting

I split the WSJ portion of Ontonotes, using sections 2-21 for training, and section 23 for testing. For all models, I used the POS tags predicted by the Stanford POS tagger (Toutanova et al., 2003) [27]. For the pipeline model and joint(+pred_span), I used MWE spans and MWE_POS tags predicted by CRF [28]. For dependency parsing, I used Redshift (Honnibal et al., 2013) for all models, with a beam size of 16 for decoding. For training, I removed non-projective dependency trees. For testing, I parsed all sentences. To evaluate parsing, I used unlabeled and labeled

[27] I used 20-way jackknifing for the training split. The test split was automatically tagged by the POS tagger trained on the training split.

[28] I used 20-way jackknifing for the training split. The test split was automatically tagged by the sequential labeler trained on the training split.

	Dependency Parsing (First tokens of MWEs)				MWE Recognition		
	Functional MWEs		NEs		Functional MWEs		NEs
Model	**UAS**	**LAS**	**UAS**	**LAS**	**FUM**	**FTM**	**FUM**
Pipeline	78.89	64.01	85.58	82.41	96.76	96.42	89.81
Joint	71.28	65.05	85.07	81.49	91.01	89.93	88.47
Joint(+dict)	79.93	73.70	85.79	82.82	**97.94**	**97.25**	90.16
Joint(+pred_span)	**81.31**	**74.74**	**85.89**	**83.23**	97.59	96.91	**91.32**

Table 7: Breakdown of experimental results by type of MWE. Note that UAS / LAS are calculated regarding the first tokens of MWEs. For NEs, the FTM is the same as the FUM because each NE always takes NNP as an MWE-level POS tag, and is not repeated.

attachment scores (UAS/LAS) [29]. To focus on dependency edges that connect inside and outside of MWEs, I measured UAS/LAS for not only the whole test set but also first tokens of gold MWEs. For the pipeline model, a parser predicts an MWE-aware dependency tree. To calculate UAS/LAS for each word belonging to MWEs, I replaced each concatenated token corresponding to an MWE with a subtree in which the first word of the MWE governs other words of the MWE (the head-initial dependency subtree).

For the joint model, I directly compared a predicted tree with the gold tree. To evaluate MWE recognition, I used the F-measure for untagged / tagged MWEs (FUM/FTM) [30]. For the pipeline model, I compared the gold MWEs with predictions by CRF. For the proposed joint model, I compared the gold MWEs with predicted MWE spans and MWE_POS tags represented as dependency labels.

[29] When calculating UAS/LAS, I removed punctuation.

[30] FUM only focuses on MWE spans, whereas FTM focuses on both MWE spans and MWE_POS tags.

4.5 Experimental Results and Discussion

I present the experimental results in Table 6. Comparing the joint model with the pipeline model, there is not much difference between these models regarding UAS / LAS for all sentences. However, the former is 2.13 / 0.48 points worse than the latter in terms of UAS / LAS regarding the first tokens of MWEs (1,269 in 34,526 tokens), and 2.37 / 2.53 points worse than the latter regarding FUM / FTM. These results suggest that the performance of the joint model with no additional features at predicting dependencies inside and around MWEs is worse than the pipeline model. One of the reasons for this is that the exploitation of head-initial structures in the joint model (Figure 18) involves the addition of MWE-specific labels. This results in an increase in the total number of dependency labels from 41 to 50. Because of this broader output space, more search errors can occur in the joint model compared with the pipeline model. Moreover, a breakdown by type of MWE (Table 7) shows that most differences in performance between these two models are related to functional MWEs. These results suggest that constraints regarding functional MWEs during parsing (4.3.2) are harmful to the joint model with no additional features in terms of its performance concerning functional MWEs.

By adding MWE-specific features to the joint model, however, I observe at least a 2.52 / 3.00 point improvement in terms of UAS / LAS regarding the first tokens of MWEs, and a 2.90 / 2.99 point improvement regarding FUM / FTM. As a result, I obtain a 1.35 / 1.28 point improvement with joint(+pred_span) compared with the pipeline model in terms of FUM / FTM. A breakdown by type of MWE shows that the addition of MWE-specific features leads to performance improvement, especially for functional MWEs (Table 7). These results suggest that MWE-specific features are effective at both MWE recognition through dependency parsing and the prediction of dependencies connecting inside and outside of MWEs.

Comparing the joint(+pred_span) with the joint(+dict), the former is 0.40 / 0.55 points better than the latter in terms of UAS / LAS regarding the first tokens of MWEs, and 0.82 / 0.82 points better than the latter regarding FUM / FTM. I can attribute this gain in performance to the additional features extracted from more accurate predictions of MWE spans and MWE_POS tags by CRF than

those by dictionary matching.

4.6 Related Work

Whereas French Treebank is available for French MWEs (Abeillé et al., 2003), there have been only limited corpora for English MWE-aware dependency parsing. Schneider et al. (2014) constructs an MWE-annotated corpus on English Web Treebank (Bies et al., 2012). However, this corpus is relatively small as training data for a parser, and its MWE annotations are not consistent with syntactic trees. By contrast, my corpus covers the whole of the WSJ portion of Ontonotes and ensures consistency between MWE annotations and parse trees.

Korkontzelos and Manandhar (2010) reports an improvement in base-phrase chunking by pre-grouping MWEs as words-with-spaces. They focus on compound nouns, adjective-noun constructions, and named entities. However, they use gold MWE spans, and this is not a realistic setting. By contrast, I use predicted MWE spans.

Three works concerned with a French MWE-aware syntactic parsing are relevant. First, Green et al. (2013) proposes a method for recognizing contiguous MWEs as a part of constituency parsing by using MWE-specific non-terminals. They investigate a CFG-based model and a model based on tree-substitution grammars. Second, Candito and Constant (2014) compares several architectures for graph-based dependency parsing and MWE recognition, in which MWE recognition is conducted before, during, and after parsing. Finally, Nasr et al. (2015) explores a joint model of MWE recognition and dependency parsing. They focus on complex function words. In terms of data representation, they adopt one similar to mine, insofar as the components of an MWE are linked by dependency edges whose labels are MWE-specific.

4.7 Summary

I constructed a corpus that ensures consistency in Ontonotes between dependency structures and English MWEs, including named entities. Furthermore, I explored models that can predict both MWE spans and an MWE-aware dependency structure. Experimental results show that by using additional MWE span features,

the joint model achieves an MWE recognition improvement of 1.35 points over the pipeline model.

5 Construction of Verbal Multiword Expression Annotated Corpus

5.1 Introduction

In previous chapters, I deal with continuous MWEs, such as functional MWEs and named entities. However, in downstream tasks, it is also important to recognize verbal MWEs (VMWEs), whose accurate recognition is challenging because VMWEs are likely to have discontinuous occurrences (e.g., **take .. off**). I show the main categories of VMWEs in Table 8. Based on this, in this chapter, I perform VMWE annotations on Ontonotes with crowdsourcing.

While dependency parsing and MWE recognition could be solved independently, dependency structures where each MWE is a syntactic unit are preferable to word-based dependency structures for downstream NLP tasks, such as semantic parsing (Chapter 3). Because MWE recognition could help syntactic parsing (Nivre and Nilsson, 2004; Eryiğit et al., 2011), several works tackle MWE-aware dependency parsing in French (Candito and Constant, 2014; Nasr et al., 2015). They use French Treebank (Abeillé et al., 2003) because of its explicit MWE annotations.

Regarding English MWEs, Schneider et al. (2014) constructs an MWE-annotated corpus on English Web Treebank (Bies et al., 2012). However, the number of VMWE occurrences (1,444) and types (1,155) in their corpus is relatively small-scale. Therefore, in this work, I conduct full-scale VMWE annotations on the Wall Street Journal (WSJ) portion of English Ontonotes (Pradhan et al., 2007), which results in 7,833 VMWE occurrences and 1,608 types. This resource enables to develop large-scale English MWE recognition and MWE-aware parsing models.

Concretely, I construct a VMWE dictionary based on the English-language Wiktionary [31]. Based on this dictionary, I collect possible VMWE occurrences on Ontonotes and filter candidates with a help of gold dependency trees. To exploit crowdsourcing, I formalize VMWE annotations as a multiword-sense disambiguation problem. This resource is available at the following URL: `https:`

[31] `https://en.wiktionary.org`

//github.com/naist-cl-parsing/Verbal-MWE-annotations.

5.2 Corpus Construction

5.2.1 Candidate Extraction

First, I construct a VMWE dictionary by extracting multiword verbs from an English part of Wiktionary [32]. I exclude auxiliary verbs and MWEs consisting of be-verbs and non-verbal components (e.g., be above, be with). As a result, I get 8,369 VMWE types.

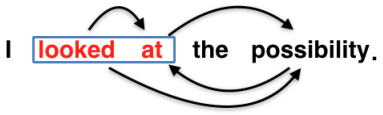


Figure 19: Dependency trees with function-head (above) and content-head schemes (below). I omit edges common in both trees. A box corresponds to a VMWE ("look at"). To filter a possible VMWE as a subtree of a dependency tree, a function-head scheme is preferable to a content-head scheme.

Second, I extract possible VMWE occurrences in 37,015 sentences of the WSJ portion of Ontonotes Release 5.0 (LDC2013T19) by using the above VMWE dictionary. I allow each VMWE to include gaps (e.g., **take .. off**), consider

Categories	**Examples**
Verb-particle constructions	pick up, take over
Prepositional verbs	look for, base on
Light verb constructions	make a decision, take a look
Verb-noun(-preposition)	take care (of)
Semi-fixed VMWEs	make one's way

Table 8: Main categories of Verbal MWEs.

[32] I select multiword entries that have "English_verbs" as categories.

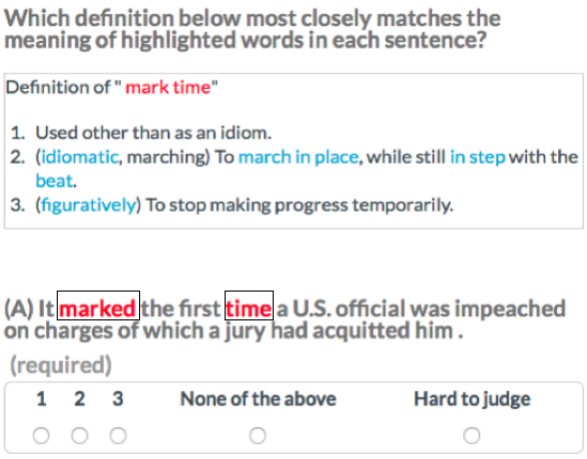


Figure 20: A screenshot of a web interface for VMWE annotations on CrowdFlower.

inflections of verbs and a variability of placeholders in semi-fixed MWEs (e.g., someone, something, one's and oneself). I exclude candidates that do not include any verbs by using gold part-of-speech information. Also, I filter out candidates that have other verbs or punctuation marks within the gaps.

Because most of the VMWEs are syntactically regular, I filter a VMWE whose components form a subtree in a Stanford basic dependency tree (de Marneffe and Manning, 2008) that is converted from a phrase structure tree given in Ontonotes. I exploit Stanford basic dependency because its function-head scheme is suitable for filtering positive occurrences of VMWEs that have a frequent POS pattern, "V IN". In many cases, a noun phrase follows this type of MWE. Therefore, in the content-head scheme like Universal Dependencies (McDonald et al., 2013), a verb of this MWE governs a head of the noun phrase, that is, such MWE does not form a subtree (Figure 19). On the contrary, such MWE corresponds to a subtree in a function-head scheme.

Regarding phrasal verbs (PVs), I perform additional filtering. In this work, I construct a VMWE-annotated corpus by extending Komai et al. (2015)'s corpus, because they have partially performed annotations of PVs on Ontonotes. For PVs that are not covered by their dictionary, I adopt the following methods: (1) I classify PVs into verb-particle constructions (VPCs) or prepositional verbs (Baldwin

et al., 2009), (2) I examine a label of a dependency edge from a verb to a particle. For VPCs, I regard a candidate as a positive VMWE occurrence iff the dependency label is "prt". For prepositional verbs, if the dependency label is "prep", and there is no gap between the verb and the particle, I regard this candidate as a positive VMWE occurrence. This is subject to rules proposed by Komai et al. (2015). Otherwise, I conduct crowdsourced annotations.

5.2.2 Large-scale Annotations of VMWEs by Crowdsourcing

Based on the above filtering, I conduct large-scale VMWE annotations on the WSJ portion of English Ontonotes by crowdsourcing whose web interface is shown in Figure 20. To exploit crowdsourcing, I formalize VMWE annotations as a multiword sense disambiguation problem. Annotators read a sentence in which some possible components of a VMWE are highlighted. They are also given possible definitions of the VMWE, extracted from an English part of Wiktionary. For each VMWE, I provide one literal sense and idiomatic senses [33]. Based on this, they are asked to determine which definition most closely matches the meaning of highlighted words in the sentence. During annotations, workers are allowed to answer that the meaning of highlighted words is not in the given senses ("None of the above"), or they are not certain of the multiword sense ("Hard to judge").

I collect crowdsourced annotations of VMWEs by using CrowdFlower [34]. I set the following requirements: (1) Annotators belong to *Level 3* contributors, who are regarded as the smallest group of most experienced, highest accuracy contributors in CrowdFlower, (2) Annotators live in countries with English as an official language, (3) Annotators accomplish a success rate higher than 70 % in answering test questions for which I give gold answers. To facilitate annotations, I provide workers with an interface to show multiple sentences (less than 6) that include possible occurrences of the same VMWE. I collect three judgments for each of 2,135 possible VMWE occurrences. Data collection costs $1,016 USD in total.

To determine whether each VMWE candidate is positive or not, I adopt the

[33] I add a definition corresponding to a literal sense if it is omitted in Wiktionary.

[34] `https://www.crowdflower.com`

following criteria:

1. If all judgments correspond to the same sense, I adopt it (67.1 %). If the sense is idiomatic, I regard this candidate as a VMWE.

2. If any judgment does not correspond to a literal sense, I regard the candidate as a positive occurrence of the VMWE (9.0 %).

3. Otherwise, I annotate the candidate manually (23.8 %).

5.2.3 Resolution of Inclusions and Overlaps

Finally, I check inclusions and overlaps between annotations by myself and those by (Komai et al., 2015), which results in 159 inclusions and 40 overlaps. Regarding inclusions, I adopt the broader MWE spans. For instance, given two MWE occurrences corresponding to "come at" and "come at a price", in which a span of the latter includes a span of the former, I leave only the latter. Concerning overlaps, I merge overlapped MWE spans if I can get a new VMWE that is in both of the following dictionaries: Cambridge Dictionary [35] and The Free Dictionary [36]. For instance, I get an occurrence of "take over the reins" by merging occurrences of "take the reins" and "take over". Also, I resolve pseudo overlaps originated from false annotations. As a result, I reduce the number of overlaps to 11 instances, which correspond to essential overlaps, such as "look back" and "look .. on .. as" in the following sentence: "He may be able to **look back on** this election **as** the high-water mark of far-left opposition.".

5.2.4 Corpus Statistics

As a result of annotations, I get 1,608 VMWE types and 7,833 instances on Ontonotes. I show histograms by the number of constituent word tokens (Table 9) and by the number of gaps (Table 10). Moreover, frequent POS patterns are shown in Table 11, in which you can see various kinds of VMWE, such as phrasal verbs (PVs), light verb constructions (LVCs), and semi-fixed MWEs. Top-3 POS patterns ("V IN", "V RP", and "V RB") correspond to PVs. Each of those includes a fair amount of discontinuous instances.

[35] `http://dictionary.cambridge.org`

[36] `http://idioms.thefreedictionary.com`

	# of constituent tokens				
	2	3	4	≥ 5	Total
VMWE instances	7,067	597	138	31	7,833
VMWE types	1,235	270	80	23	1,608

Table 9: Corpus statistics. I show VMWE types and instances by the number of constituent word tokens.

# of gaps	0	1	2
VMWE instances	6,855	968	10

Table 10: A histogram of VMWE instances by the number of gaps.

5.3 Related Work

I introduce several MWE-annotated corpora. First, French Treebank (Abeillé et al., 2003) is often used as a dataset for French MWE-aware dependency parsing (Candito and Constant, 2014) because of its explicit MWE annotations. It consists of phrase structure trees, augmented with morphological information and functional annotations of verbal dependents. Second, Vincze (2012) provides English-Hungarian parallel corpus annotated for LVCs, which belong to VMWEs. Their corpus contains 703 LVCs in Hungarian and 727 in English on 14,261 sentence alignment units, taken from economiclegal texts and literature. Recently, PARSEME organizes a shared task on automatic identification of verbal MWEs (Savary and Ramisch, 2017). They provide annotation guidelines and annotated corpora of 5.5 million tokens and 60,000 VMWE annotations for 18 languages. Note that their corpora do not support English at edition 1.0.

Regarding English MWEs, Shigeto et al. (2013) first constructs an MWE dictionary by extracting functional MWEs [37] from the English-language Wiktionary, and classifies their occurrences in Ontonotes into either MWE or literal usage.

[37]Here, I refer to MWEs that function either as prepositions, conjunctions, determiners, pronouns, adverbs, auxiliary verbs, to-infinitives, or interjections as functional MWEs.

In Kato et al. (2017), I integrate annotations of these functional MWEs and named entities (NEs) [38] into phrase structures by establishing MWEs as subtrees. After that, I exploit this dataset for experiments of English MWE-aware dependency parsing (Chapter 4).

5.4 Summary

In this work, I conduct large-scale annotations of English VMWEs on the Wall Street Journal portion of Ontonotes. Based on a VMWE dictionary extracted from English Wiktionary, I collect possible VMWE occurrences on Ontonotes, and filter candidates with a help of gold dependency trees. To take advantage of crowdsourcing, I formalize annotations of VMWEs as a multiword-sense disambiguation problem. My future work could involve the followings:

1. I plan to integrate the above VMWE annotations into annotations for functional MWEs and named entities on Ontonotes by Kato et al. (2016) and Kato et al. (2017). This will help to develop models for MWE recognition and dependency parsing that are aware of various kinds of English MWEs.

2. I get VMWE occurrences on Ontonotes for only 1,608 out of 8,369 types in the VMWE dictionary. Therefore, I plan to explore VMWE occurrences on the larger corpus, such as the Annotated English Gigaword treebank [39].

[38] The NE annotations are given by Ontonotes.

[39] `http://catalog.ldc.upenn.edu/LDC2012T21`

POS patterns	Continuous VMWEs	Discontinuous VMWEs	Frequent VMWEs
V IN	3,071	260	base on : 142 look for : 86 focus on : 77 go to : 70 account for : 69
V RP	2,081	229	set up : 62 take over : 49 point out : 47 turn out : 43 pick up : 39
V RB	547	116	go back : 17 come back : 17 do well : 15 go down : 13 go ahead : 13
V NN	280	167	take place : 41 do business : 27 take effect : 26 take steps : 24 have time : 22
V DT NN	114	45	take a look : 13 make a decision : 8 pave the way : 5 lay the groundwork : 5
V RP IN	98	4	come up with : 20 make up for : 12 keep up with : 8 live up to : 7 add up to : 5
V JJ	77	11	make sure : 14 go wrong : 8 go public : 6 keep quiet : 5 make much : 4
V IN NN	56	26	have in mind : 8 take into account : 7 set in motion : 5 sign into law : 5
V V	47	32	be called : 34 be had : 5 have got : 4 make known : 4 let know : 4
V PRP	77	0	make it : 16 have it : 10 buy it : 9 move it : 5
V PRP$ NN	49	1	have one's way : 5 run one's course : 4 make one's way : 3 read someone's lips : 3

Table 11: Corpus statistics based on POS patterns (at least 50 occurrences only).

6 Joint Analysis of Continuous Multiword Expression-Aware Dependency Parsing and Discontinuous Multiword Expression Recognition

6.1 Introduction

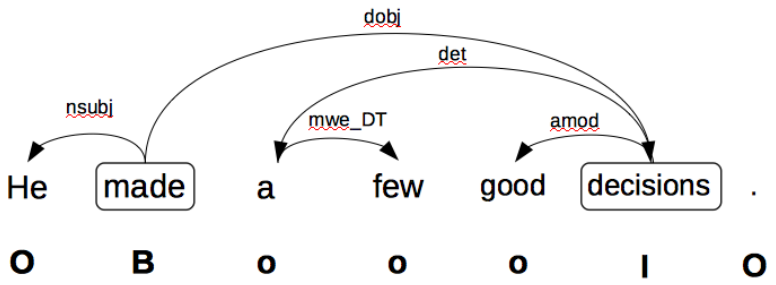


Figure 21: A sentence that includes both continuous ("a few") and discontinuous MWEs ("made .. decisions"). For this sentence, a model predicts both a continuous MWE-aware dependency tree (shown in the upper half of this figure) and a verbal MWE (VMWE). In the bottom half of this figure, I show a sequence of extended BIO tags (Schneider et al., 2014), which can represent gaps between components of a discontinuous MWE with "o" (small-o).

In downstream tasks that require the automated understanding of the meaning of the texts, it is important to recognize not only continuous MWEs [40] but also VMWEs such as phrasal verbs, which are likely to have discontinuous occurrences. Because dependency information, specifically, a dependency edge from a verb to a particle or a direct object is expected to be effective in VMWE recognition, in this chapter, I address the task to predict both continuous MWE-aware dependency trees and VMWEs (Figure 21) (Chapter 6.3).

Regarding continuous MWE-aware dependency parsing, I explore the following three models:

[40] Here, I define continuous MWEs as MWEs that always have continuous occurrences, such as functional and adjective MWEs.

1. A pipeline model of continuous MWE recognition (CMWER) and MWE-aware dependency parsing. In this model, I formalize CMWER as sequential labeling, merge a predicted MWE span to a single node and perform dependency parsing.

2. Single-task(parser), which is a model to predict a word-based dependency tree that encodes MWE spans and MWE-level POS-tags as dependency labels.

3. The hierarchical multi-task learning (HMTL) (Sanh et al., 2018) of CMWER and Single-task(parser).

In the HMTL-based model, an output from a low-level task encoder is fed into a high-level task encoder. The motivation to use this model is the following: continuous MWE-aware dependency parsing could be decomposed to CMWER and dependency parsing in which a CMWE is treated as a syntactic unit. Therefore, in the HMTL-based model, the high-level task, that is, Single-task(parser) is able to utilize features captured by the low-level task (CMWER). These features are expected to improve the performances of Single-task(parser).

Another contribution in this chapter is the development of a continuous MWE-aware dependency corpus. In Chapter 3, I developed a dependency corpus that is aware of functional MWEs. However, there are other kinds of continuous MWEs, such as adjective MWEs and compound nouns. Because compound nouns are syntactically compositional and highly productive, annotations of compound nouns are much more expensive than those of adjective MWEs. Therefore, in this chapter, I conduct adjective MWE annotations on Ontonotes to broaden the coverage of continuous MWEs. After that, I modify phrase structure trees in Ontonotes so that a span of a functional or adjective MWE comprises a subtree in a phrase structure tree. Finally, I perform tree-to-dependency conversion and develop a large-scale dependency corpus that is aware of both functional and adjective MWEs (Chapter 6.2).

Experiments in continuous MWE-aware dependency corpus built on Ontonotes 5.0 (Chapter 6.2) show that the pipeline and HMTL-based models are 1.7 points better than Single-task(parser) in terms of F1 in CMWER. Concerning MWE-aware dependency parsing, all models lead to comparable UAS in the whole test

set and sentences including MWEs. If I focus on the first tokens of gold MWEs, Single-task(parser) and the HMTL-based model perform at least 1.4 points better than the pipeline model in terms of UAS.

Concerning VMWE recognition (VMWER), I formalize this task as sequential labeling by using extended BIO tagging scheme (Schneider et al., 2014), which is able to treat gaps between components of a VMWE (e.g., *pick .. up*). I explore the following two models:

1. Single-task (VMWE), which is the sequential labeler for VMWE recognition.

2. The HMTL of CMWER, Single-task(parser), and Single-task (VMWE).

I investigate the above HMTL-based model because dependency information, specifically, a dependency edge from a verb to a particle or a direct object is expected to be effective in VMWE recognition.

As a dataset, I use VMWE annotations (Kato et al., 2018) on Ontonotes 5.0, which I described in Chapter 5. Experimental results show that the above HMTL-based model performs 1.3 points better than Single-task (VMWE) in terms of F1.

6.2 Construction of Continuous Multiword Expression-Aware Dependency Corpus

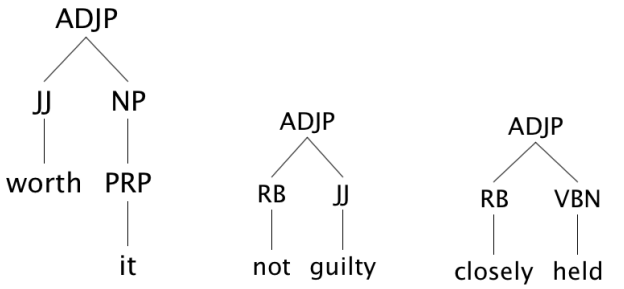
Here, I describe the construction of an English dependency corpus that is aware of both adjective and functional MWEs. This corpus is based on an English treebank and MWE annotations. Regarding the English treebank, I use the Wall Street Journal (WSJ) portion of Ontonotes 5.0 (Pradhan et al., 2007). Concerning MWE annotations, I perform annotations of adjective MWEs and merge them with annotations of functional MWEs by Shigeto et al. (2013).

6.2.1 Annotations of Functional MWEs

Regarding functional MWEs, I adopt annotations by Shigeto et al. (2013). Note that their target of functional MWE annotations, Penn Treebank (Marcus et al., 1994) includes my target of adjective MWE annotations, the WSJ portion of Ontonotes 5.0. They perform MWE annotations in the following four steps.

First, they construct an MWE dictionary by extracting functional MWEs from English Wiktionary [41]. Second, they collect MWE candidates on Penn Treebank with a dictionary matching. Third, for each type of functional MWEs, they cluster its candidate occurrences according to their phrase structures. Finally, they classify each cluster into MWE or literal usages.

6.2.2 Annotations of Adjective MWEs



(a) An example of an MWE usage.

(b) An example of a literal usage.

(c) Another example of a literal usage.

Figure 22: Examples in which an MWE span corresponds to a phrase structure subtree that has ADJP as a root node. Based on PCFG rule probabilities, (a) is annotated as an MWE usage, and (b) and (c) are classified into literal usages.

I perform annotations of adjective MWEs in the following three steps.

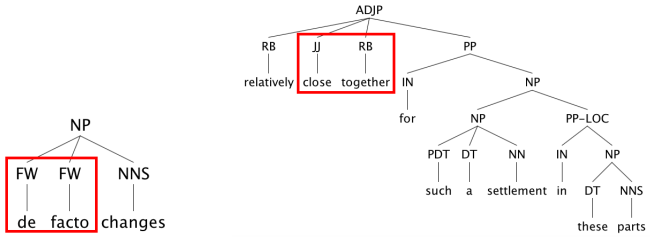
First, I extract adjective MWEs from English Wiktionary and construct an MWE dictionary, which has 2,869 MWE types.

Second, I conduct a dictionary matching in the WSJ portion of Ontonotes 5.0. As a result, 304 types of adjective MWEs appear in this corpus at least once.

Third, according to whether each MWE has at least one semantic non-compositional sense, I divide the above 304 types of adjective MWEs into two groups, which I describe below. I consider a sense of an MWE is semantic non-compositional if Wiktionary treats the sense as idiomatic or figurative.

1. Concerning an adjective MWE that has at least one idiomatic or figurative sense (69 types, 380 candidate occurrences), I classify each candidate into

[41] `https://en.wiktionary.org`



(a) An example of an MWE usage.

(b) An example of a literal usage.

Figure 23: Examples in which an MWE span does not correspond to a phrase structure subtree. Based on PCFG rule probabilities, (a) and (b) are annotated as MWE and literal usages, respectively.

MWE or literal usages with word sense disambiguation (WSD). I perform WSD in the following procedures. If a characteristic syntactic pattern accompanies an idiomatic or figurative sense, I perform WSD based on phrase structures and POS tags of components of a candidate MWE [42]. Otherwise, I perform WSD manually by myself. Based on the results of WSD, I treat each candidate as an MWE usage if the meaning of the candidate corresponds to an idiomatic or figurative sense. Otherwise, I classify the candidate into a literal usage.

2. Regarding an adjective MWE that does not have any idiomatic or figurative senses (235 types, 1,034 candidate occurrences), I conduct annotations in the following two steps. First, I classify candidates into two groups (case(A) and (B) to be described) based on whether a candidate MWE span comprises a subtree in a phrase structure tree. Second, I classify candidates in each group into MWE or literal usages according to syntactic non-compositionality. Concretely, for a given candidate MWE, if a PCFG rule suffices the following condition, I classify a candidate into an MWE usage.

$$\Pr(ADJP \rightarrow t_1..t_n) < 0.0008 \quad (2)$$

[42]For instance, in the most of adjective MWE usages of "to go", only these two tokens ("to go") form a verb phrase (e.g., I have three more years *to go*). Hence, if a phrase subtree that minimally covers "to go" includes other tokens, I treat this instance as a literal usage.

Here, $t_1..t_n$ is a POS tag sequence of components of a candidate MWE. Hereafter, I call the above condition $Cond_{PCFG}$. I induce PCFG rules from the training split (sections 02-21) of the WSJ portion of Ontonotes 5.0.

Case (A): If there is a non-terminal symbol (hereafter X) that corresponds to a candidate MWE span (457 candidate occurrences), I perform MWE annotations in the following way.

i). If X is ADJP (adjective phrase) (Figure 22), I classify a candidate into MWE (Figure 22) or literal usages (Figure 22 (b) and (c)) based on a probability of a PCFG rule, as I describe above.

ii). If X is PP (prepositional phrase), I classify a candidate into an MWE usage if the PP functions as an adjective phrase and $Cond_{PCFG}$ holds for it.

iii). If X is neither ADJP nor PP, I treat MWE candidates as literal usages.

Case (B): If a candidate MWE span does not correspond to any subtree in a phrase structure (577 candidates), I classify a candidate into an MWE usage if it functions as an adjective phrase and it suffices $Cond_{PCFG}$ (Figure 23).

POS tag sequences	No. of instances	Examples
RB JJ	44	dead last, legally binding
IN IN NN	19	up to date, out of business
FW FW	17	de facto, ad hoc
IN JJ	13	on alert, as much
IN NN	11	in place, in question

Table 12: The statistics of adjective MWE annotations. I show POS tag sequences that appear more than 10 times.

As a result, I acquire 83 types, 198 occurrences of adjective MWEs in the

MWE-level POS tags	RB	IN	JJ	others
MWE Instances	3,794	2,409	198	655
MWE Types	377	91	83	31

Table 13: Corpus statistics of functional and adjective MWE annotations.

WSJ portion of Ontonotes 5.0. I show the corpus statistics regarding POS tag sequences of components of adjective MWEs in Table 12.

6.2.3 Construction of MWE-aware Dependency Corpus

I construct a dependency corpus that is aware of functional and adjective MWEs in the following three steps.

(A) A resolution of conflicts between functional and adjective MWE annotations.

(B) A resolution of conflicts between MWE annotations and phrase structures.

(C) A conversion from phrase structures to dependency trees.

6.2.3.1 A resolution of conflicts between functional and adjective MWE annotations

First, I explore conflicts between functional and adjective MWE annotations. As a result, I find that some adjective MWE spans include functional MWE spans. For instance, an adjective MWE span, "out of business" includes a functional MWE span, "out of" in the following sentence:

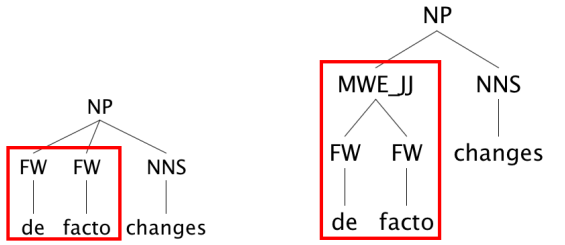
.. and putting unprofitable state-owned companies **out of business**.

In such instance, I resolve a conflict between these two annotations by leaving the broader MWE span. I show the corpus statistics of functional and adjective MWE annotations after this resolution in Table 13.

6.2.3.2 A resolution of conflicts between MWE annotations and phrase structures

Next, I resolve conflicts between MWE annotations and phrase structures. When some annotated MWE span does not correspond to a phrase structure subtree, I call this situation a conflict between an MWE annotation and a phrase structure. Because conflicts between functional MWE annotations and phrase structures have already been resolved in the corpus by Kato et al. (2016), I extend it by dealing with conflicts between phrase structures and adjective MWE annotations, which I perform additionally in this work.

Based on methods I described in Chapter 3, I classify adjective MWE spans into the following three patterns according to consistency between an MWE span and a phrase structure (Finkel and Manning, 2009; Kato et al., 2016).



(a) A phrase structure before a modification. A rectangle shows an MWE span.

(b) A phrase structure after a modification. An MWE ("de facto") is grouped as a subtree.

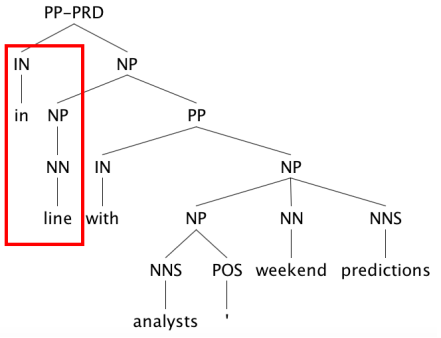
Figure 24: A modification of a phrase structure in a case of "multiple contiguous children".

(1) A phrase structure subtree (Figure 22a)

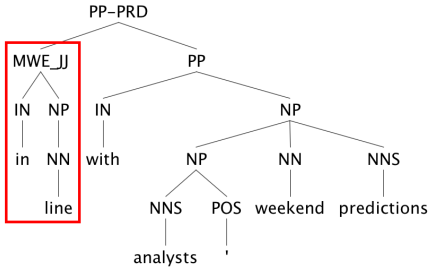
If an MWE span corresponds to a phrase structure subtree, I need not modify this constituency tree.

(2) Multiple contiguous children (Figure 24)

If an MWE span corresponds to multiple contiguous children of a subtree (Finkel and Manning, 2009; Kato et al., 2016), I insert an MWE-specific

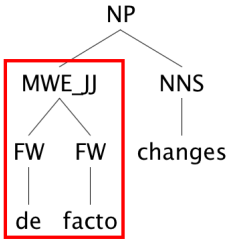


(a) A phrase structure before a modification. A rectangle shows an MWE span.

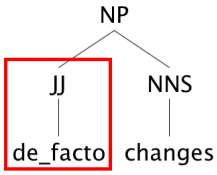


(b) A phrase structure after a modification. An MWE ("in line") is grouped as a subtree.

Figure 25: A modification of a phrase structure in a case of "crossing brackets".



(a) Before a merge



(b) After a merge

Figure 26: A step of merging components of an MWE into a single node.

non-terminal [43] that governs the MWE span. Note that I do not require manual annotations for this case.

(3) Crossing brackets (Figure 25)

If MWE spans have crossing brackets with the spans in the parse tree, I establish each MWE as a subtree by the following way. As a typical example, a subtree that includes an adjective MWE span could be a PP comprised of a preposition (IN) and a noun phrase (NP). This NP consists of a nested NP and a nested PP (Figure 25a). In such instances, an adjective MWE span precedes a nested PP. In this case, I insert an MWE-specific non-terminal that governs an adjective MWE span and move the nested PP

[43] Format of the MWE-specific non-terminal is "MWE_<MWE-level POS-tag>" (e.g., "MWE_JJ").

to the main PP (Figure 25b).

6.2.3.3 A conversion from phrase structures to dependency trees

Finally, I perform tree-to-dependency conversion. As a preprocessing step, I first replace a subtree corresponding to an MWE by a preterminal node and its child node (Figure 26). The preterminal node has an MWE-level POS tag. The child node is created by joining all components of the MWE with underscores (words-with-spaces). After that, I convert phrase structures to Stanford basic dependency trees (de Marneffe and Manning, 2008) [44].

6.3 Models for Continuous MWE-aware Dependency Parsing and Verbal MWE Recognition

Here, I describe the models for continuous MWE-aware dependency parsing and verbal MWE recognition.

6.3.1 Continuous MWE-aware Dependency Parsing

For continuous MWE-aware dependency parsing, I investigate the following three models:

1. A pipeline model of continuous MWE recognition (CMWER) with the sequential labeler, and MWE-aware dependency parsing, which treats an MWE as a syntactic unit (Figure 27).

2. Single-task(parser), which is a model to predict a word-based dependency tree that encodes MWE spans and MWE-level POS-tags as dependency labels (a head-initial dependency tree) (Figure 28).

3. The hierarchical multi-task learning (HMTL) (Sanh et al., 2018) of CMWER and Single-task(parser) (Figure 29).

[44] I use Stanford CoreNLP Ver.3.9.1. As a command-line option, I use the following: -basic -keepPunct -conllx -originalDependencies. Details are described in the following site:`https://nlp.stanford.edu/software/stanford-dependencies.html`.

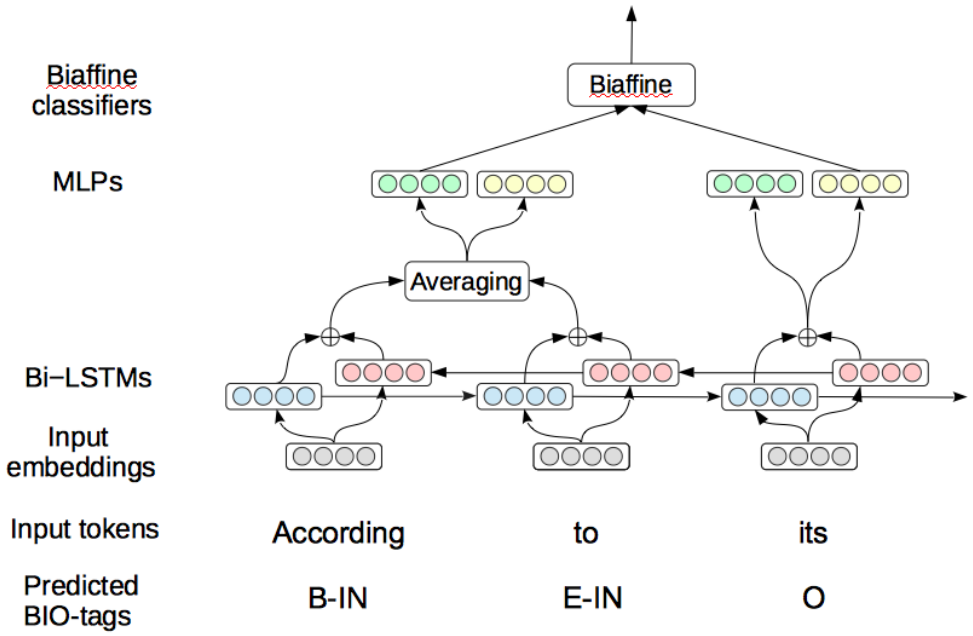


Figure 27: An architecture of the pipeline model. In this model, I get an average of hidden states corresponding to an MWE span predicted with the sequential labeler, and use it as an input to MLPs whose outputs are fed into biaffine classifiers.

Input features for the above models are concatenations of word embeddings, ELMo embeddings (Peters et al., 2018), and character-level word representations (Kim et al., 2016).

In the pipeline model, I use the bi-LSTM-CNNs-CRF model (Ma and Hovy, 2016) for CMWER and Deep biaffine parser (Dozat and Manning, 2017) for MWE-aware dependency parsing. To calculate a hidden state of an MWE span, I average output vectors that correspond to tokens belonging to an MWE span predicted by the sequential labeler, from a bi-LSTM network (Figure 27). This hidden vector of an MWE is fed into MLPs in Deep biaffine parser. Hence, an output from the parser is a dependency tree in which a predicted MWE is treated as a syntactic unit. Note that I utilize gold MWE spans in training to make it possible for the parser to predict a gold dependency tree. At inference time, I

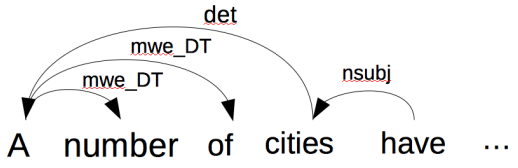


Figure 28: An example of a head-initial dependency tree, in which a span of an MWE ("a number of") and an MWE-level POS-tag are encoded as dependency labels (mwe_DT).

exploit MWE spans predicted by the sequential labeler.

In Single-task(parser), I utilize Deep biaffine parser to predict a head-initial dependency tree. Because this model predicts a head-initial dependency tree, in which MWE spans and MWE-level POS-tags are represented as dependency labels, this model could mitigate error propagation, which the pipeline model could suffer from.

Finally, I describe the motivation to use the HMTL-based model. Because continuous MWE-aware dependency parsing could be decomposed to CMWER and dependency parsing in which a CMWE is treated as a syntactic unit, features captured by the sequential labeler for CMWER are expected to improve performances of Single-task(parser). Therefore, I adopt the HMTL-based model in which CMWER and Single-task(parser) are treated as low and high-level tasks, respectively. In the HMTL-based model, inputs to the high-level task encoder are both outputs from the low-level task encoder and token representations of an input sentence (shortcut connections (Sanh et al., 2018)).

6.3.2 Verbal MWE Recognition

With respect to VMWE recognition (VMWER), I formalize this task as sequential labeling by using extended BIO tagging scheme (Schneider et al., 2014), which is able to treat gaps between components of a VMWE (e.g., *pick .. up*). I explore the following two models:

1. Single-task (VMWE), which is the sequential labeler for VMWE recognition. Concretely, I use the bi-LSTM-CNNs-CRF model (Ma and Hovy,

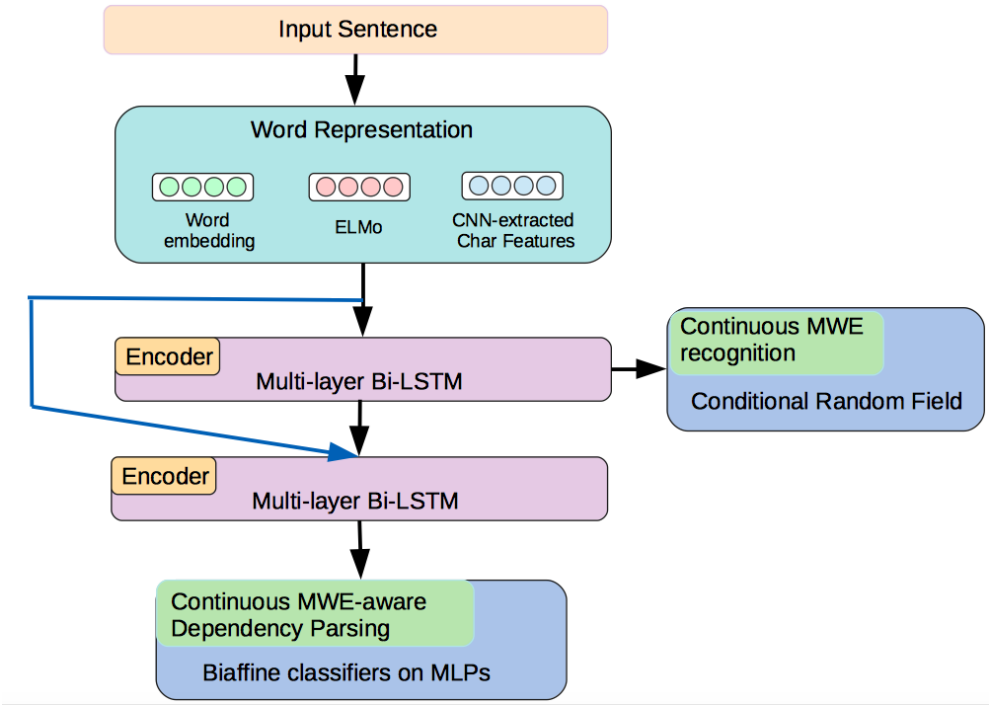


Figure 29: An architecture of the hierarchical multi-task learning (HMTL) of continuous MWE recognition (CMWER) and dependency parsing which predicts head-initial dependency trees (Single-task(parser)).

2016).

2. The HMTL of CMWER, Single-task(parser), and Single-task (VMWE).

I investigate the above HMTL-based model because dependency information, specifically, a dependency edge from a verb to a particle or a direct object is expected to be effective in VMWE recognition. In this HMTL-based model, the encoder of Single-task (VMWE) can utilize features captured by the encoder of Single-task(parser). For comparison, I explore the following two HMTL-based models:

1. the HMTL of CMWER and Single-task (VMWE).
2. the HMTL of Single-task(parser) and Single-task (VMWE).

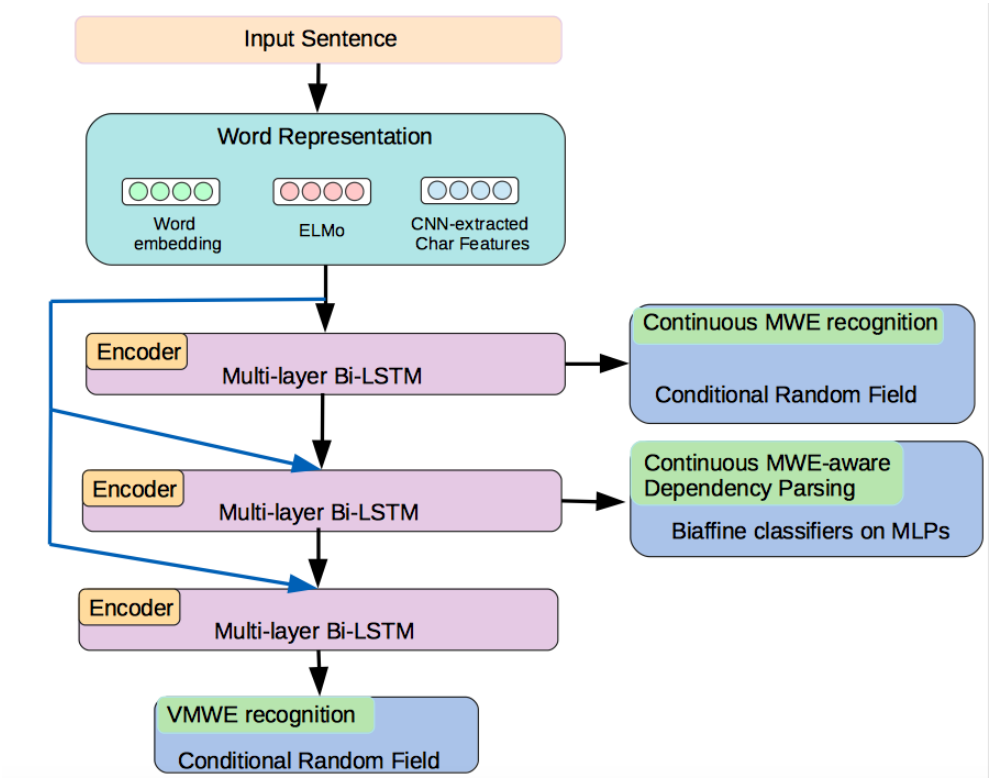


Figure 30: An architecture of HMTL of continuous MWE recognition (CMWER), dependency parsing that predicts head-initial dependency trees (Single-task(parser)), and Single-task (VMWE).

6.4 Experiments with Models for Continuous MWE-aware Dependency Parsing and Verbal MWE Recognition

Here, I describe experiments of models for continuous MWE-aware dependency parsing and verbal MWE recognition.

6.4.1 Experimental Setup and Implementation Details

As a dataset, I use a continuous MWE-aware dependency corpus (Chapter 6.2) and VMWE annotations (Chapter 5) (Kato et al., 2018) on Ontonotes 5.0. I split the WSJ portion of Ontonotes 5.0, using sections 2-21 for training, section 22 for development, and section 23 for testing, respectively.

Within the above VMWE annotations, I find a small number of sentences in which multiple VMWE annotations partially overlap with each other. Because it is difficult to formalize VMWE recognition in these sentences as sequential labeling, I perform experiments after removing 13 and 2 sentences with overlapped VMWE annotations from the training and test splits, respectively.

During the training of an HMTL-based model, a task is randomly sampled and a minibatch of a dataset attached to the task is also randomly sampled. The sampling probability for a task is proportional to the relative size of a dataset for the task (proportional sampling) [45]. As a criterion of convergence, I adopt Early stopping (Prechelt, 1998), in which training would be stopped if the validation metrics do not improve within the fixed number of epochs, which is a hyper-parameter.

As base implementations of bi-LSTM-CNNs-CRF and Deep biaffine parser, I use the AllenNLP library (Gardner et al., 2018). I initialize word embeddings with Glove (Pennington et al., 2014). Concerning ELMo representations, I utilize the bi-LSTM network pre-trained with One Billon Word Benchmark corpus (Chelba et al., 2013). In the bi-LSTM-CNNs-CRF models used in CMWER and VMWER, I apply dropout to inputs and outputs of the bi-LSTM networks. In Deep biaffine parser, I use dropout in inputs and outputs of the bi-LSTM networks and outputs from Multi-Layer Perceptrons (MLPs) for dependency heads and dependency labels.

Downsampling of negative instances in the sequential labeling

In continuous MWE-aware dependency corpus, many sentences have no MWEs (e.g., 1370 in 1640 sentences in the test set), which corresponds to sequences made of only "O" (outside) tags. Hereafter, I define negative and positive instances as a sentence having no MWEs and a sentence having at least one MWE, respectively. To mitigate class imbalance (Leevy et al., 2018), in training of the sequential labeler for recognition of CMWEs or VMWEs, I perform downsampling of negative

[45] In my experiments, the sizes of datasets for tasks in HMTL are almost the same with each other, however, the VMWE recognition task has less training instances than those of the other two tasks (CMWER and continuos MWE-aware dependency parsing) because I remove sentences that have partially overlapped VMWEs, as I described above.

instances if number of negative instances are more than positive instances in a mini-batch. The purpose of this downsampling is to make the number of negative instances that contribute to the loss function, same as those of positive instances.

6.4.2 Hyper-parameters and model selections

Hyper-parameter	Value
[Input features]	
Word embeddings	Glove 6B (100 dimensions)
Character-based representations	16 dimensions
Window size	3
No. of filters	64
[Training]	
Optimizer	Adam
Learning rate	0.001
Epochs	50
Patience (used in early stopping)	10
[Continuous MWE recognition]	
Hidden Units of LSTMs	64
No. of LSTM Layers	2
Dropout ratio	0.2
Mini-batch Size	32
[Continuous MWE-aware dependency parsing]	
Hidden Units of LSTMs	256
No. of LSTM Layers	3
Dropout ratio	0.33
Hidden Units of MLPs (for dependency arcs)	500
Hidden Units of MLPs (for dependency labels)	100
Mini-batch Size	100
[VMWE recognition]	
Hidden Units of LSTMs	64
No. of LSTM Layers	2
Dropout ratio	0.2
Mini-batch Size	32

Table 14: Hyper-parameters used in the experiments.

I show hyper-parameters used in experiments in Table 14. Regarding CMWER, I select a model based on F1-scores of the development set. Concerning VMWER, and HMTL-based models that include VMWER, I perform a model selection based on F1-scores for VMWEs of the development set. With respect to CMWE-aware dependency parsing and other HMTL-based models, I select a model based on labeled attachment scores (LAS) of the development set.

6.4.3 Evaluation measures

To evaluate continuous MWE recognition, I use the F-measure for untagged / tagged MWEs (FUM / FTM) [46]. Regarding models that predict head-initial dependency trees, I calculate FUM / FTM by using predicted MWE spans and MWE-level POS tags represented in head-initial dependency trees.

Concerning an evaluation measure of VMWE recognition, I use FUM. Each predicted VMWE is represented as token indices because a VMWE could have a gap between its elements.

Regarding continuous MWE-aware dependency parsing (CMWE-DP), I use unlabeled and labeled attachment scores (UAS / LAS) [47]. For the pipeline model of CMWER and CMWE-DP, each MWE span predicted by CMWER is merged into a single token in the first stage. After that, the parser predicts an CMWE-aware dependency tree in the second stage. Hence, in evaluation, I convert each merged token into a head-initial subtree and calculate UAS and LAS.

[46] FUM only focuses on MWE spans, whereas FTM focuses on both MWE spans and MWE-level POS tags.

[47] When calculating UAS / LAS, I remove punctuation.

6.4.4 Experimental Results and Discussion

	CMWE-DP						**CMWE**		**VMWE**
	All sentences		**Sentences including MWEs**		**First words of gold MWEs**				
Model	**UAS**	**LAS**	**UAS**	**LAS**	**UAS**	**LAS**	**FUM**	**FTM**	**FUM**
Single task (parser)	**95.78**	94.23	**95.03**	92.83	**85.85**	77.21	93.25	92.56	
Pipeline (CMWE + parser)	95.76	**94.26**	94.83	92.82	84.21	76.19	**95.01**	**94.80**	
HMTL (CMWE + parser)	95.69	94.20	94.89	**92.86**	85.64	**77.75**	95.00	94.38	
Single-task (VMWE)									84.28
HMTL (CMWE + VMWE)							94.99	94.58	83.12
HMTL (parser + VMWE)	95.19	93.45	94.12	91.63	82.24	73.20	91.11	89.72	84.57
HMTL (CMWE + parser + VMWE)	95.02	93.31	94.09	91.87	83.26	75.24	94.35	93.74	**85.62**

Table 15: Experimental results for the test set regarding continuous MWE-aware dependency parsing (CMWE-DP) and VMWE recognition (VMWER). The result is an average of five independent trials.

I show experimental results for the test set in Table 15. First, regarding CMWER, Pipeline(CMWE + parser), HMTL (CMWE + parser), and HMTL (CMWE + VMWE) show similar FUMs, which are 1.7 points better than an FUM of Single-task (parser). These results suggest that sequential labeling or the HMTL-based models are more suitable to recognize CMWEs than CMWER through CMWE-aware dependency parsing.

MWE-level POS tags	**IN**		**RB**		**DT**		**JJ**	
# of MWE instances	93		167		18		11	
Model	**UAS**	**LAS**	**UAS**	**LAS**	**UAS**	**LAS**	**UAS**	**LAS**
Single task (parser)	79.79	70.11	**87.19**	79.16	**100.00**	93.33	**87.27**	**70.91**
Pipeline (continuous MWE + parser)	77.42	68.60	85.99	78.68	98.89	90.00	83.64	69.09
HMTL (continuous MWE + parser)	80.43	69.89	86.83	**80.60**	**100.00**	**94.44**	81.82	63.64
HMTL (parser + VMWE)	78.92	71.40	81.79	72.22	**100.00**	92.22	80.00	60.00
HMTL (continuous MWE + parser + VMWE)	**80.86**	**73.12**	82.39	74.49	**100.00**	92.22	81.82	65.46

Table 16: Experimental results for the test set regarding continuous MWE-aware dependency parsing with respect to first words of gold MWEs. I omit results for MWEs that have PRP as MWE-level POS tags, which occur 5 times in the test set, because UAS and LAS of these MWEs are 100% for all models.

Second, I describe the results of CMWE-aware dependency parsing. For the whole test set and sentences including MWEs (270 in 1640 sentences), all models show similar unlabeled attachment scores (UASs). Because the number of tokens belonging to MWEs corresponds to about 1.9 % of that of the whole test set, multiple models could show similar performances for the whole test set, even though these models perform differently in terms of UAS regarding dependency edges that are inside of CMWEs or that connect inside and outside of CMWEs. Actually, if I focus on UAS of first words of gold MWEs, HMTL (CMWE + parser) performs 1.4 points better than Pipeline(CMWE + parser). HMTL (CMWE + parser) exploits outputs from the encoder of the CMWE recognizer as input features for the encoder of the dependency parser, instead of using predictions of CMWER deterministically. Hence, HMTL (CMWE + parser) mitigates error propagation from the CMWE recognizer, which is consistent with the above results. Besides, HMTL (CMWE + parser) and Single-task (parser) show similar UASs. Considering the former is better than the latter in terms of CMWER, the HMTL of CMWER and the head-initial dependency parser operates success-

fully. To compare performances of models in more detail, I show experimental results of continuous MWE-aware dependency parsing concerning first words of gold MWEs in Table 16. These results suggest that HMTL-based models including CMWER are effective to predict dependency heads of first words of MWEs that have IN (prepositions or subordinate conjunctions) as MWE-level POS tags. Regarding the first words of adverbial MWEs, Single task (parser) performs the best in terms of UAS and leads to at least 4.8 points better than HMTL-based models including VMWER. Note that most of the gold MWEs in the test set also appear in the training set (285 in 294 functional or adjective MWEs). Furthermore, I find that PP-attachment often leads to parsing errors. For instance, Single task (parser) incorrectly predicts "building" as a dependency head of the first word of the gold MWE, "around the world" instead of the correct head, "bureaus" in the following sentence: "It focused on building up its news bureaus **around the world**, so as events took place it could go live quicker and longer than other networks."

Finally, concerning VMWE recognition, HMTL (CMWE + parser + VMWE) performs 1.3 points better than Single-task (VMWE) in terms of FUM. Considering CMWE spans are represented in head-initial dependency trees, HMTL (parser + VMWE) and HMTL (CMWE + parser + VMWE) receive the same supervised signals. In spite of this, the latter performs 1.0 points better than the former in terms of FUM. This result suggests that it is more effective for VMWE recognition models to exploit not only features captured by the dependency parser, which predicts CMWE spans through parsing but also features acquired by the sequential labeler for CMWE recognition. Besides, HMTL (CMWE + parser + VMWE) is 2.5 points better than HMTL (CMWE + VMWE) in terms of FUM. This result infers that the syntactic information captured by the CMWE-aware dependency parser is effective for VMWE recognition. Furthermore, I investigate the performances of HMTL (CMWE + parser + VMWE) in terms of unknown VMWE recognition. Among 278 types of VMWEs that appear in the test set, 76 types of VMWEs do not appear in the training set, which I call unknown VMWEs. The detailed analysis shows that recalls of HMTL(CMWE + parser + VMWE) regarding unknown and known VMWEs are 56.25 and 91.87, respec-

tively [48]. Hence, it is worthwhile to investigate the effects of additional features based on a VMWE dictionary that I develop in Chapter 5.2.1.

6.5 Related Work

First, I describe MWE-aware syntactic corpora. Whereas French Treebank (Abeillé et al., 2003) is often used as a dataset for French MWE-aware dependency parsing (Candito and Constant, 2014) because of its explicit MWE annotations, there have been only limited corpora for English MWE-aware dependency parsing. Schneider et al. (2014) performs MWE annotations on English Web Treebank (Bies et al., 2012). However, their MWE annotations do not guarantee that an MWE span corresponds to a phrase structure subtree. Besides, their corpus consists of about 3,800 sentences, which is relatively small as the training data for a dependency parser. By contrast, my corpus ensures that an MWE span comprises a subtree in a phrase structure tree (section 6.2.3), and it covers the whole of the WSJ portion of Ontonotes 5.0, which consists of about 37,000 sentences.

Next, I introduce several works concerned with an MWE-aware syntactic parsing. First, Green (2013) (Green et al., 2013) introduces MWE-specific nonterminals and proposes a method to recognize contiguous MWEs as a part of constituency parsing. They explore two models based on context-free grammars and tree substitution grammars, respectively. Second, Candito and Constant (2014) addresses joint dependency parsing and French MWE recognition. For MWEs that do not have syntactic non-compositionality, they adopt alternative representations instead of head-initial dependency trees (Figure 28). Their motivation is to represent a syntactic structure within a syntactically regular MWE. Their models focus on not only dependency parsing and MWE identification, but also predictions of internal structures of syntactically regular MWEs. In their models, MWE recognition is conducted before, during, or after parsing. Third, Constant and Nivre (2016) addresses a joint prediction of the following two structures: a dependency tree and a forest of lexical units including MWEs. These two structures share lexical elements, that is, both tokens and syntactically non-compositional MWEs. In the latter structure, each MWE is represented by a constituency-like tree, which is able to consider nested or discontinuous MWEs.

[48] This is the average of five independent experiments.

They use a transition-based system that is an extension of a standard dependency parser. They adopt classical transition-based parsing features, i.e., combinations of linguistic attributes of nodes on the stacks and the buffer, processed subtrees, and transition history. While their model predicts both lexical units and a dependency tree that is aware of syntactically non-compositional MWEs, this work tackles both VMWE recognition and continuous MWE-aware dependency parsing. Because my motivation is to develop an MWE-aware dependency corpus that is easy to use in downstream tasks that require an automated understanding of the meaning of the texts, I integrate various continuous MWEs into dependency structures, which are not limited to syntactically non-compositional MWEs. Finally, Kato et al. (2017, 2016) ensures that a span of a functional MWE or a named entity (NE) comprises a subtree in a phrase structure tree in Ontonotes. Based on this dataset, I perform tree-to-dependency conversion to get an English dependency corpus that is aware of both NEs and functional MWEs (Chapter 4). As is the case with the work in this chapter, I explore the following two models in Chapter 4: (1) a pipeline model of the recognizer of NEs and functional MWEs, and the dependency parser that treats an NE or an MWE as a unit, (2) a joint model that performs word-based dependency parsing to predict a head-initial dependency tree. I use Conditional Random Fields (CRF) (Lafferty et al., 2001) as the sequential labeler and an arc-eager transition system (Nivre, 2003) as the dependency parser in Chapter 4. This work is based on Kato et al. (2017), and perform additional annotations for adjective MWEs to cover broader categories of MWEs. Moreover, I address not only continuous MWE-aware dependency parsing but also discontinuous MWE recognition.

Regarding VMWE-annotated corpora, PARSEME organizes shared tasks on automatic identification of VMWEs (Ramisch and Cordeiro, 2018; Savary and Ramisch, 2017). They provide annotation guidelines and annotated corpora of 79,326 VMWE annotations for 20 languages. Several works tackle this shared task with neural network-based approaches. Klyueva et al. (2017) formalizes VMWE identification as sequential labeling and use a bi-directional recurrent neural network with gated-recurrent units (GRUs) (Cho et al., 2014). Berk et al. (2018) utilizes the bi-LSTM-CRF model for VMWE recognition.

6.6 Summary

In this chapter, I address continuous MWE-aware dependency parsing and VMWE recognition. My contribution is two folds. First, I construct a dependency corpus that is aware of both functional and adjective MWEs. Second, I design several models and compare their performances in terms of continuous MWE (CMWE)-aware dependency parsing and MWE recognition. Experimental results show that the hierarchical multi-task learning (HMTL)–based model of the CMWE recognizer, the head-initial dependency parser, and the VMWE recognizer is effective in VMWE recognition.

7 Conclusion

In this thesis, I develop continuous MWE-aware dependency corpora and VMWE annotations. Furthermore, I address the following two tasks: continuous MWE-aware dependency parsing and discontinuous MWE recognition.

My contribution is the followings:

(1) A formalization to develop MWE-aware dependency corpus

The main motivation for MWE-aware dependency trees comes from the fact that most of existing English treebanks do not guarantee that an MWE span corresponds to a phrase structure subtree. Hence, one could not acquire MWE-aware dependency trees by simply merging components of each MWE in dependency trees converted from phrase structure trees.

To deal with this problem, given a phrase structure tree and MWE annotations, I formalized procedures to ensure that an MWE span corresponds to a phrase structure subtree by modifying phrase structure trees (Chapter 3).

(2) Development of continuous MWE-aware dependency corpora

Based on the above procedures, I developed MWE-aware dependency corpus ver.1.0 (LDC2017T01) and 2.0 (LDC2017T16) in Ontonotes 5.0 (Pradhan et al., 2007). The former is aware of functional MWEs (Chapter 3), and the latter is aware of NEs and functional MWEs (Chapter 4).

(3) VMWE annotations on Ontonotes

In downstream tasks, it is important to recognize not only continuous MWEs but also verbal MWEs (VMWEs), which is likely to have discontinuous occurrences (e.g., **take .. off**). Therefore, I performed VMWE annotations on Ontonotes 5.0 with crowdsourcing (Chapter 5). To exploit crowdsourcing, I formalized VMWE annotations as a multiword-sense disambiguation problem. My VMWE annotations on Ontonotes 5.0 are publicly available [49].

[49] `https://github.com/naist-cl-parsing/Verbal-MWE-annotations`

(4) Continuous MWE-aware dependency parsing and VMWE recognition

Based on the intuition that dependency information is expected to be effective in VMWE recognition, I tackled the task to predict both continuous MWE-aware dependency trees and VMWEs (Chapter 6). By experiments with continuous MWE-aware dependency corpus and VMWE annotations on Ontonotes 5.0, I demonstrated that the effectiveness of a model based on the hierarchical multi-task learning (HMTL) (Sanh et al., 2018) of the following three tasks: continuous MWE recognition, a prediction of head-initial dependency trees, and VMWE recognition. Besides, I perform adjective MWE annotations to broaden the coverage of continuous MWEs in Ontonotes.

As future work, I plan the followings:

(1) The extension of coverage of MWEs

Even though I integrate functional and adjective MWEs into dependency structures, several kinds of MWEs are not covered. Hence, I plan to integrate VMWEs, auxiliary verbal MWEs (aux-VMWEs), and semi-fixed MWEs (Constant and Nivre, 2016; Morimoto et al., 2016) into dependency trees.

(2) Use of pre-trained language models for MWE recognition

Very recently, it is demonstrated that contextualized token representations acquired with pre-trained neural language models (NLMs) boost performances of various downstream tasks (Peters et al., 2018; Devlin et al., 2018). However, it is still unclear whether pre-trained LMs are useful in MWE recognition. Specifically, I would like to explore a capacity of NLMs to recognize syntactic and/or semantic non-compositionality of MWEs in an unsupervised or weakly supervised way, under task settings in which an MWE dictionary is given while MWE annotations are not available.

(3) Cross-lingual transfer learning of VMWE classification

In recent years, PARSEME organizes shared tasks on automatic identification of VMWEs (Ramisch and Cordeiro, 2018; Savary and Ramisch, 2017). Datasets of these tasks consist of VMWE annotations for 20 languages.

Based on the above datasets, I plan to design models for cross-lingual transfer learning of VMWE classification (Taslimipoor et al., 2019). Concretely, I assume the followings: (1) VMWE dictionaries and annotations for two similar languages, such as English and German are given, (2) The scale of VMWE annotations for a source language is larger than that for a target language, (3) I can train cross-lingual word representations (Ruder et al., 2017; Eriguchi et al., 2018) of these two languages with word-level or sentence-level alignments. The former could be obtained with a bilingual dictionary, while the latter could be acquired with parallel translation data.

Given the above assumptions, I plan to perform cross-lingual transfer learning of VMWE classification in the following procedures: (1) I extract candidates of VMWE instances in each language by matching sentences with a VMWE dictionary considering inflections and gaps between components of a potential VMWE, (2) I train a model that performs a binary classification of potential VMWE instances into MWE or literal usages in a source language, (3) I additionally train this model with the training data from a target language. This model is expected to mitigate limited training data for the target language.

References

Anne Abeillé, Lionel Clément, and François Toussenel. 2003. *Building a Treebank for French*, pages 165–187. Springer Netherlands, Dordrecht.

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. *ArXiv*, abs/1603.06042.

Timothy Baldwin, Valia Kordoni, and Aline Villavicencio. 2009. Prepositions in applications: A survey and introduction to the special issue. *Computational Linguistics*, 35:119–149.

Gözde Berk, Berna Erden, and Tunga Güngör. 2018. Deep-BGT at PARSEME shared task 2018: Bidirectional LSTM-CRF model for verbal multiword expression identification. In *Proceedings of the Joint Workshop on Linguistic Annotation, Multiword Expressions and Constructions (LAW-MWE-CxG-2018)*, pages 248–253, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Ann Bies, Justin Mott, Colin Warner, and Seth Kulick. 2012. English web treebank. *Technical Report LDC2012T13, Linguistic Data Consortium, Philadelphia, Pennsylvania, USA.*

Jari Björne, Filip Ginter, and Tapio Salakoski. 2017. Epe 2017: The biomedical event extraction downstream application. In *Proceedings of the 2017 Shared Task on Extrinsic Parser Evaluation (EPE 2017) at the Fourth International Conference on Dependency Linguistics (Depling 2017) and the 15th International Conference on Parsing Technologies (IWTP 2017)*, pages 17–24. Association for Computational Linguistics.

Marie Candito and Matthieu Constant. 2014. Strategies for contiguous multiword expression analysis and dependency parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 743–753. Association for Computational Linguistics.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, and Phillipp Koehn. 2013. One billion word benchmark for measuring progress in statistical language modeling. In *INTERSPEECH*.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750, Doha, Qatar. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. In *SSST@EMNLP*.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *ACL*.

Matthieu Constant and Joakim Nivre. 2016. A transition-based system for joint lexical and syntactic analysis. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 161–171. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. *ArXiv*, abs/1611.01734.

Timothy Dozat, Peng Qi, and Christopher D. Manning. 2017. Stanford's graph-based neural dependency parser at the CoNLL 2017 shared task. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 20–30, Vancouver, Canada. Association for Computational Linguistics.

Jason M. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th Conference on Computational Linguistics - Volume 1*, COLING '96, pages 340–345, Stroudsburg, PA, USA. Association for Computational Linguistics.

Akiko Eriguchi, Melvin Johnson, Orhan Firat, Hideto Kazawa, and Wolfgang Macherey. 2018. Zero-shot cross-lingual classification using multilingual neural machine translation. *ArXiv*.

Gülşen Eryiğit, Tugay İlbay, and Ozan Arkan Can. 2011. Multiword expressions in statistical dependency parsing. In *Proceedings of the Second Workshop on Statistical Parsing of Morphologically Rich Languages*, pages 45–55, Dublin, Ireland. Association for Computational Linguistics.

Jenny Rose Finkel and Christopher D. Manning. 2009. Joint parsing and named entity recognition. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 326–334, Boulder, Colorado. Association for Computational Linguistics.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew E. Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2018. Allennlp: A deep semantic natural language processing platform. *ArXiv*, abs/1803.07640.

Spence Green, Marie-Catherine de Marneffe, John Bauer, and Christopher D. Manning. 2011. Multiword expression identification with tree substitution grammars: A parsing tour de force with french. In *EMNLP*.

Spence Green, Marie-Catherine de Marneffe, and Christopher D Manning. 2013. Parsing models for identifying multiword expressions. *Computational Linguistics*, 39(1):195–227.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Matthew Honnibal, Yoav Goldberg, and Mark Johnson. 2013. *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, chapter A Non-Monotonic Arc-Eager Transition System for Dependency Parsing. Association for Computational Linguistics.

Akihiko Kato, Hiroyuki Shindo, and Yuji Matsumoto. 2016. Construction of an English dependency corpus incorporating compound function words. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1667–1671, Portorož, Slovenia. European Language Resources Association (ELRA).

Akihiko Kato, Hiroyuki Shindo, and Yuji Matsumoto. 2017. English multiword expression-aware dependency parsing including named entities. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 427–432, Vancouver, Canada. Association for Computational Linguistics.

Akihiko Kato, Hiroyuki Shindo, and Yuji Matsumoto. 2018. Construction of large-scale English verbal multiword expression annotated corpus. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Languages Resources Association (ELRA).

Yoon Kim, Yacine Jernite, David A Sontag, and Alexander M. Rush. 2016. Character-aware neural language models. In *AAAI*.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional lstm feature representations. *Transactions of the Association for Computational Linguistics*, 4:313–327.

Natalia Klyueva, Antoine Doucet, and Milan Straka. 2017. Neural networks for multi-word expression detection. In *Proceedings of the 13th Workshop on Multiword Expressions (MWE 2017)*, pages 60–65, Valencia, Spain. Association for Computational Linguistics.

Masayuki Komai, Hiroyuki Shindo, and Yuji Matsumoto. 2015. An efficient annotation for phrasal verbs using dependency information. In *Proceedings of the 29th Pacific Asia Conference on Language, Information and Computation: Posters*, pages 125–131.

Ioannis Korkontzelos and Suresh Manandhar. 2010. Can recognising multiword expressions improve shallow parsing? In *Human Language Technologies: The*

2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, pages 636–644, Los Angeles, California. Association for Computational Linguistics.

Sandra Kubler, Ryan McDonald, Joakim Nivre, and Graeme Hirst. 2009. *Dependency Parsing*. Morgan and Claypool Publishers.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Joffrey L. Leevy, Taghi M. Khoshgoftaar, Richard A. Bauder, and Naeem Seliya. 2018. A survey on addressing high-class imbalance in big data. *Journal of Big Data*, 5:1–30.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *ACL*.

Xuezhe Ma and Eduard H. Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *ArXiv*, abs/1603.01354.

Xuezhe Ma, Zecong Hu, Jingzhou Liu, Nanyun Peng, Graham Neubig, and Eduard H. Hovy. 2018. Stack-pointer networks for dependency parsing. In *ACL*.

Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The penn treebank: Annotating predicate argument structure. In *Proceedings of the Workshop on Human Language Technology*, HLT '94, pages 114–119, Stroudsburg, PA, USA. Association for Computational Linguistics.

Marie-Catherine de Marneffe and Christopher D. Manning. 2008. The stanford typed dependencies representation. In *CFCFPE at COLING*.

Ryan McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar

Täckström, Claudia Bedini, Núria Bertomeu Castelló, and Jungmee Lee. 2013. Universal dependency annotation for multilingual parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 92–97. Association for Computational Linguistics.

Ryan T. McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. 2005. Non-projective dependency parsing using spanning tree algorithms. In *HLT/EMNLP*.

Ayaka Morimoto, Akifumi Yoshimoto, Akihiko Kato, Hiroyuki Shindo, and Yuji Matsumoto. 2016. Identification of flexible multiword expressions with the help of dependency structure annotation. In *Proceedings of the Workshop on Grammar and Lexicon: interactions and interfaces (GramLex)*, pages 102–109, Osaka, Japan. The COLING 2016 Organizing Committee.

Courtney Napoles, Matthew R. Gormley, and Benjamin Van Durme. 2012. Annotated gigaword. In *AKBC-WEKEX@NAACL-HLT*.

Alexis Nasr, Carlos Ramisch, José Deulofeu, and André Valli. 2015. Joint dependency parsing and multiword expression tokenization. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1116–1126, Beijing, China. Association for Computational Linguistics.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT*, pages 149–160.

Joakim Nivre. 2004. Incrementality in deterministic dependency parsing.

Joakim Nivre and Jens Nilsson. 2004. Multiword units in syntactic parsing. In *Workshop on Methodologies and Evaluation of Multiword Units in Real-World Applications*, pages 39–46, Lisbon, Portugal.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference*

on Empirical Methods in Natural Language Processing (EMNLP), pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

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

Sameer S. Pradhan, Eduard Hovy, Mitch Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2007. Ontonotes: A unified relational semantic representation. In *Proceedings of the International Conference on Semantic Computing*, ICSC '07, pages 517–526, Washington, DC, USA. IEEE Computer Society.

Lutz Prechelt. 1998. Early stopping-but when? In *Neural Networks: Tricks of the Trade, This Book is an Outgrowth of a 1996 NIPS Workshop*, pages 55–69, London, UK, UK. Springer-Verlag.

Carlos Ramisch and Silvio Ricardo Cordeiro. 2018. Edition 1.1 of the parseme shared task on automatic identification of verbal multiword expressions. In *Proceedings of the Joint Workshop on Linguistic Annotation, Multiword Expressions and Constructions*, pages 222–240, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Sebastian Ruder, Ivan Vuli'c, and Anders Sogaard. 2017. A survey of cross-lingual word embedding models.

Victor Sanh, Thomas Wolf, and Sebastian Ruder. 2018. A hierarchical multi-task approach for learning embeddings from semantic tasks. *arXiv preprint arXiv:1811.06031*.

Agata Savary and Carlos Ramisch. 2017. The PARSEME shared task on automatic identification of verbal multiword expressions. In *Proceedings of the 13th Workshop on Multiword Expressions (MWE 2017)*, pages 31–47, Valencia, Spain. Association for Computational Linguistics.

Nathan Schneider, Spencer Onuffer, Nora Kazour, Emily Danchik, Michael T. Mordowanec, Henrietta Conrad, and Noah A. Smith. 2014. Comprehensive

annotation of multiword expressions in a social web corpus. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 455–461, Reykjavik, Iceland. European Language Resources Association (ELRA). ACL Anthology Identifier: L14-1433.

Yutaro Shigeto, Ai Azuma, Sorami Hisamoto, Shuhei Kondo, Tomoya Kouse, Keisuke Sakaguchi, Akifumi Yoshimoto, Frances Yung, and Yuji Matsumoto. 2013. *Proceedings of the 9th Workshop on Multiword Expressions*, chapter Construction of English MWE Dictionary and its Application to POS Tagging. Association for Computational Linguistics.

Shiva Taslimipoor, Omid Rohanian, and Le An Ha. 2019. Cross-lingual transfer learning and multitask learning for capturing multiword expressions. In *Proceedings of the Joint Workshop on Multiword Expressions and WordNet (MWE-WN 2019)*, pages 155–161, Florence, Italy. Association for Computational Linguistics.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*.

Veronika Vincze. 2012. Light verb constructions in the szegedparalellfx english-hungarian parallel corpus. In *LREC*.

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 188–193. Association for Computational Linguistics.

加藤 明彦，進藤 裕之，松本 裕治. 2019. 複単語表現を考慮した依存構造コーパスの構築と解析. **自然言語処理**, 26(4):663–688.

List of Major Publications

Journal Paper

1. 加藤 明彦，進藤 裕之，松本 裕治. 2019. 複単語表現を考慮した依存構造コーパスの構築と解析. **自然言語処理**, 26(4):663–688

Conference Paper (referred)

1. Akihiko Kato, Hiroyuki Shindo, and Yuji Matsumoto. 2016. Construction of an English dependency corpus incorporating compound function words. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1667–1671, Portorož, Slovenia. European Language Resources Association (ELRA)

2. Akihiko Kato, Hiroyuki Shindo, and Yuji Matsumoto. 2017. English multiword expression-aware dependency parsing including named entities. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 427–432, Vancouver, Canada. Association for Computational Linguistics

3. Akihiko Kato, Hiroyuki Shindo, and Yuji Matsumoto. 2018. Construction of large-scale English verbal multiword expression annotated corpus. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Languages Resources Association (ELRA)

List of Linguistic Resources

MWE-Aware English Dependency Corpus

This dependency corpus (`https://catalog.ldc.upenn.edu/LDC2017T01`) is aware of functional MWEs (Chapter 3).

MWE-Aware English Dependency Corpus 2.0

This dependency corpus (`https://catalog.ldc.upenn.edu/LDC2017T16`) is aware of named entities and functional MWEs (Chapter 4).

An English verbal MWE annotations on Ontonotes 5.0

These verbal MWE annotations (Chapter 5) are available at: `https://github.com/naist-cl-parsing/Verbal-MWE-annotations`.