Semantic Structure Analysis of Noun Phrases using Abstract Meaning Representation

Yuichiro Sawai **Hiroyuki Shindo** **Yuji Matsumoto**
Graduate School of Information Science
Nara Institute of Science and Technology
8916-5 Takayama, Ikoma, Nara, 630-0192, Japan
{sawai.yuichiro.sn0,shindo,matsu}@is.naist.jp

Abstract

We propose a method for semantic structure analysis of noun phrases using Abstract Meaning Representation (AMR). AMR is a graph representation for the meaning of a sentence, in which noun phrases (NPs) are manually annotated with internal structure and semantic relations. We extract NPs from the AMR corpus and construct a data set of NP semantic structures. We also propose a transition-based algorithm which jointly identifies both the nodes in a semantic structure tree and semantic relations between them. Compared to the baseline, our method improves the performance of NP semantic structure analysis by 2.7 points, while further incorporating external dictionary boosts the performance by 7.1 points.

1 Introduction

Semantic structure analysis of noun phrases (NPs) is an important research topic, which is beneficial for various NLP tasks, such as machine translation and question answering (Nakov and Hearst, 2013; Nakov, 2013). Among the previous works on NP analysis are internal NP structure analysis (Vadas and Curran, 2007; Vadas and Curran, 2008), noun-noun relation analysis of noun compounds (Girju et al., 2005; Tratz and Hovy, 2010; Kim and Baldwin, 2013), and predicate-argument analysis of noun compounds (Lapata, 2002).

The goal of internal NP structure analysis is to assign bracket information inside an NP (e.g., *(lung cancer) deaths* indicates that the phrase *lung cancer* modifies the head *deaths*). In noun-noun relation analysis, the goal is to assign one of the predefined semantic relations to a noun compound consisting of two nouns (e.g., assigning a relation *purpose* to a noun compound *cooking pot*, meaning that *pot* is used for *cooking*). On the other hand, in predicate-argument analysis, the goal is to decide whether the modifier noun is the subject or the object of the head noun (e.g., *car* is the object of *love* in *car lover*, while *child* is the subject of *behave* in *child behavior*).

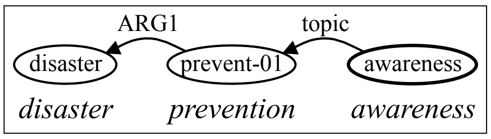

Figure 1: *disaster prevention awareness* in AMR. Predicate-argument relation *ARG1*, noun-noun relation *topic*, and internal structure *(disaster prevention) awareness* are expressed.

Previous NP researches have mainly focused on these different subproblems of NP analysis using different data sets, rather than targeting general NPs simultaneously. However, these subproblems of NP analysis tend to be highly intertwined when processing texts. For the purpose of tackling the task of combined NP analysis, we make use of the Abstract Meaning Representation (AMR) corpus.

AMR is a formalism of sentence semantic structure by directed, acyclic, and rooted graphs, in which semantic relations such as predicate-argument relations and noun-noun relations are expressed. In this paper, we extract substructures corresponding to NPs (shown in Figure 1) from the AMR Bank[1], and create a data set of NP semantic structures. In general, AMR substructures are graphs. However, since we found out that NPs mostly form trees rather than graphs in the AMR Bank, we can assume that AMR substructures corresponding to NPs are trees. Thus, we define our task as predicting the AMR tree structure, given a sequence of words in an NP.

The previous method for AMR parsing takes a

[1]http://amr.isi.edu/

Train	Dev	Test
3504	463	398

Table 1: Statistics of the extracted NP data

two-step approach: first identifying distinct concepts (nodes) in the AMR graph, then defining the dependency relations between those concepts (Flanigan et al., 2014). In the concept identification step, unlike POS tagging, one word is sometimes assigned with more than one concept, and the number of possible concepts is far more than the number of possible parts-of-speech. As the concept identification accuracy remains low, such a pipeline method suffers from error propagation, thus resulting in a suboptimal AMR parsing performance.

To solve this problem, we extend a transition-based dependency parsing algorithm, and propose a novel algorithm which jointly identifies the concepts and the relations in AMR trees. Compared to the baseline, our method improves the performance of AMR analysis of NP semantic structures by 2.7 points, and using an external dictionary further boosts the performance by 7.1 points.

2 Abstract Meaning Representation

2.1 Extraction of NPs

We extract substructures (subtrees) corresponding to NPs from the AMR Bank (LDC2014T12). In the AMR Bank, there is no alignment between the words and the concepts (nodes) in the AMR graphs. We obtain this alignment by using the rule-based alignment tool by Flanigan et al. (2014). Then, we use the Stanford Parser (Klein and Manning, 2003) to obtain constituency trees, and extract NPs that contain more than one noun and are not included by another NP. We exclude NPs that contain named entities, because they would require various kinds of manually crafted rules for each type of named entity. We also exclude NPs that contain possessive pronouns or conjunctions, which prove problematic for the alignment tool. Table 1 shows the statistics of the extracted NP data.

2.2 Previous Method for AMR Analysis

We adopt the method proposed by Flanigan et al. (2014) as our baseline, which is a two-step pipeline method of concept identification step and

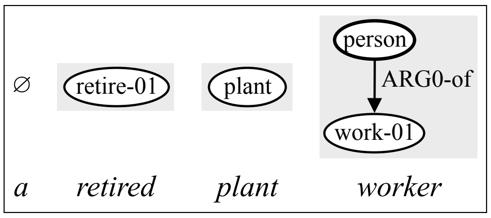

Figure 2: Concept identification step of (Flanigan et al., 2014) for *a retired plant worker*. ∅ denotes an empty concept.

relation identification step. Their method is designed for parsing sentences into AMR, but here, we use this method for parsing NPs.

In their method, concept identification is formulated as a sequence labeling problem (Janssen and Limnios, 1999) and solved by the Viterbi algorithm. Spans of words in the input sentence are labeled with concept subgraphs. Figure 2 illustrates the concept identification step for an NP *a retired plant worker*.

After the concepts have been identified, these concepts are fixed, and the dependency relations between them are identified by an algorithm that finds the maximum spanning connected subgraph (Chu and Liu, 1965), which is similar to the maximum spanning tree (MST) algorithm used for dependency parsing (McDonald et al., 2005).

They report that using gold concepts yields much better performance, implying that joint identification of concepts and relations can be helpful.

3 Proposed Method

In this paper, we propose a novel approach for mapping the word sequence in an NP to an AMR tree, where the concepts (nodes) corresponding to the words and the dependency relations between those concepts must be identified. We extend the arc-standard algorithm by Nivre (2004) for AMR parsing, and propose a transition-based algorithm which jointly identifies concepts and dependency

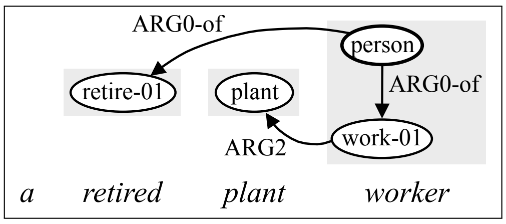

Figure 4: *a retired plant worker* in AMR

	Previous action	σ_1	σ_0	β	R
0	(initial state)			[a retired plant worker]	$\varnothing$
1	SHIFT(EMPTY(a))		$\varnothing$	[retired plant worker]	$\varnothing$
2	EMPTY-REDUCE			[retired plant worker]	$\varnothing$
3	SHIFT(DICT$_{\text{PRED}}$(retired))		retire-01	[plant worker]	$\varnothing$
4	SHIFT(LEMMA(plant))	retire-01	plant	[worker]	$\varnothing$
5	SHIFT(KNOWN(worker))	plant	$\text{person} \xrightarrow{\text{ARG0-of}} \text{work-01}$	[ ]	$\varnothing$
6	LEFT-REDUCE(ARG2, n_{child})	retire-01	$\text{person} \xrightarrow{\text{ARG0-of}} \text{work-01}$	[ ]	$\{(\text{work-01}) \xrightarrow{\text{ARG2}} (\text{plant})\}$
7	LEFT-REDUCE(ARG0-of, n_{root})		$\text{person} \xrightarrow{\text{ARG0-of}} \text{work-01}$	[ ]	$\{(\text{work-01}) \xrightarrow{\text{ARG2}} (\text{plant}),$ $(\text{person}) \xrightarrow{\text{ARG0-of}} (\text{retire-01})\}$

Figure 3: Derivation of an AMR tree for *a retired plant worker* (σ_0 and σ_1 denote the top and the second top of the stack, respectively.)

Action	Current state	Next state
SHIFT$(c(w_i))$	$(\sigma, [w_i\vert\beta], R)$	$([\sigma\vert c(w_i)], \beta, R)$
LEFT-REDUCE(r, n)	$([\sigma\vert c_i\vert c_j], \beta, R)$	$([\sigma\vert c_j], \beta, R \cup \{n_{\text{root}}(c_i) \xleftarrow{r} n(c_j)\})$
RIGHT-REDUCE(r, n)	$([\sigma\vert c_i\vert c_j], \beta, R)$	$([\sigma\vert c_i], \beta, R \cup \{n(c_i) \xrightarrow{r} n_{\text{root}}(c_j)\})$
EMPTY-REDUCE	$([\sigma\vert\phi], \beta, R)$	(σ, β, R)

Table 2: Definitions of the actions

relations. Our algorithm is similar to (Hatori et al., 2011), in which they perform POS tagging and dependency parsing jointly by assigning a POS tag to a word when performing SHIFT, but differs in that, unlike POS tagging, one word is sometimes assigned with more than one concept. In our algorithm, the input words are stored in the buffer and the identified concepts are stored in the stack. SHIFT identifies a concept subtree associated with the top word in the buffer. REDUCE identifies the dependency relation between the top two concept subtrees in the stack. Figure 3 illustrates the process of deriving an AMR tree for *a retired plant worker*, and Figure 4 shows the resulting AMR tree.

Table 2 shows the definition of each action and state transition. A state is a triple (σ, β, R), where σ is a stack of identified concept subtrees, β is a buffer of words, and R is a set of identified relations. SHIFT$(c(w_i))$ extracts the top word w_i in the buffer, generates a concept subtree $c(w_i)$ from w_i, and pushes $c(w_i)$ onto the stack. The concept subtree $c(w_i)$ is generated from w_i by using one of the rules defined in Table 3. LEFT-REDUCE(r, n) pops the top two subtrees c_i, c_j from the stack, identifies the relation r between the root $n_{\text{root}}(c_i)$ of c_i and the node $n(c_j)$ in c_j, adds r to R, and pushes c_j back onto the stack. Here, n denotes a mapping from a subtree to its specific node, which allows for attachment to an arbitrary concept in a subtree. Since the sizes of the subtrees were at most two in our data set, $n \in \{n_{\text{root}}, n_{\text{child}}\}$, where n_{root} is a mapping from a subtree to its root, and n_{child} is a mapping from a subtree to the direct child of its root. RIGHT-REDUCE(r, n) is defined in the same manner. EMPTY-REDUCE removes an empty subtree $\varnothing$ at the top of the stack. EMPTY-REDUCE is always performed immediately after SHIFT$(\varnothing)$ generates an empty subtree $\varnothing$. In the initial state, the stack σ is empty, the buffer β contains all the words in the NP, and the set of the identified relations R is empty. In the terminal state, the buffer β is empty and the stack σ contains only one subtree. The resulting AMR tree is obtained by connecting all the subtrees generated by the SHIFT actions with all the relations in R.

The previous method could not generate unseen concepts in the training data, leading to low recall in concept identification. In contrast, our method defines five rules (Table 3), three of which

Rule	Concept subtree to generate	Example of subtree generation
EMPTY	an empty concept subtree ∅	fighters → ∅
KNOWN	a subtree aligning to the word in the training data	fighters → (person) $\xrightarrow{\text{ARG0-of}}$ (fight-01) | ...
LEMMA	a subtree with the lemma of the word as the only concept	fighters → (fighter)
DICT$_{\text{PRED}}$	a subtree with a predicate form of the word as the only concept	fighters → (fight-01) | (fight-02) | ...
DICT$_{\text{NOUN}}$	a subtree with a noun form of the word as the only concept	fighters → (fight)

Table 3: Rules for generating a concept subtree (Vertical lines indicate multiple candidate subtrees.)

(LEMMA, DICT$_{\text{PRED}}$, and DICT$_{\text{NOUN}}$) allow for generation of unseen concepts from any word.

3.1 Features

The feature set $\phi(s, a)$ for the current state s and the next action a is the direct product (all-vs-all combinations from each set) of the feature set $\phi_{state}(s)$ for the current state and the feature set $\phi_{action}(s, a)$ for the next action.

$$\phi(s, a) = \phi_{state}(s) \times \phi_{action}(s, a)$$

$\phi_{state}(s)$ is the union of the feature sets defined in Table 4, where $w(c)$ denotes the word from which the subtree c was generated.

Table 5 shows the feature set $\phi_{action}((\sigma, [w_i|\beta], R), a)$ for each action a, where $rule(w_i, c)$ is a function that returns the rule which generated the subtree c from the top word w_i in the buffer. In order to allow different SHIFT actions and different LEFT-RIGHT/REDUCE actions to partially share features, Table 5 defines features of different granularities for each action. For example, although SHIFT((run-01)) and SHIFT((sleep-01)) are different actions, they share the features "S", "S"∘"DICT$_{\text{PRED}}$" because they share the generation rule DICT$_{\text{PRED}}$.

Name	Definition
LEM	$\{w(\sigma_1).\text{lem}, w(\sigma_0).\text{lem}, \beta_0.\text{lem},$ $w(\sigma_1).\text{lem} \circ w(\sigma_0).\text{lem}, w(\sigma_0).\text{lem} \circ \beta_0.\text{lem}\}$
SUF	$\{w(\sigma_1).\text{suf}, w(\sigma_0).\text{suf}, \beta_0.\text{suf},$ $w(\sigma_1).\text{suf} \circ w(\sigma_0).\text{suf}, w(\sigma_0).\text{suf} \circ \beta_0.\text{suf}\}$
POS	$\{w(\sigma_1).\text{pos}, w(\sigma_0).\text{pos}, \beta_0.\text{pos},$ $w(\sigma_1).\text{pos} \circ w(\sigma_0).\text{pos}, w(\sigma_0).\text{pos} \circ \beta_0.\text{pos}\}$
DEP	$\{w(\sigma_1).\text{dep}, w(\sigma_0).\text{dep}, \beta_0.\text{dep},$ $w(\sigma_1).\text{dep} \circ w(\sigma_0).\text{dep}, w(\sigma_0).\text{dep} \circ \beta_0.\text{dep}\}$
HEAD	$\{w(\sigma_1).\text{off}, w(\sigma_0).\text{off}, \beta_0.\text{off},$ $w(\sigma_1).\text{off} \circ w(\sigma_0).\text{off}, w(\sigma_0).\text{off} \circ \beta_0.\text{off}\}$
ROOT	$\{n_{\text{root}}(\sigma_1), n_{\text{root}}(\sigma_0), n_{\text{root}}(\sigma_1) \circ n_{\text{root}}(\sigma_0)\}$
MID	all words between $w(\sigma_1)$ and $w(\sigma_0)$ ∪ all words between $w(\sigma_0)$ and β_0

Table 4: Feature sets for the state (.lem is the lemma, .suf is the prefix of length 3, .pos is the part-of-speech, .dep is the dependency label to the parent word, .off is the offset to the parent word, and ∘ denotes concatenation of features.)

Action a	$\phi_{action}((\sigma, [w_i\|\beta], R), a)$
SHIFT(c)	{"S", "S" ∘ $rule(w_i, c)$, "S" ∘ $rule(w_i, c) \circ c$}
LEFT-REDUCE(r, n)	{"L-R", "L-R" ∘ r, "L-R" ∘ $r \circ n$}
RIGHT-REDUCE(r, n)	{"R-R", "R-R" ∘ r, "R-R" ∘ $r \circ n$}
EMPTY-REDUCE	{"E-R"}

Table 5: Feature sets for the action

4 Experiments

We conduct an experiment using our NP data set (Table 1). We use the implementation [2] of (Flanigan et al., 2014) as our baseline. For the baseline, we use the features of the default settings.

The method by Flanigan et al. (2014) can only generate the concepts that appear in the training data. On the other hand, our method can generate concepts that do not appear in the training data using the concept generation rules LEMMA, DICT$_{\text{PRED}}$, and DICT$_{\text{NOUN}}$ in Table 3. For a fair comparison, first, we only use the rules EMPTY and KNOWN. Then, we add the rule LEMMA, which can generate a concept of the lemma of the word. Finally, we add the rules DICT$_{\text{PRED}}$ and DICT$_{\text{NOUN}}$. These two rules need conversion from nouns and adjectives to their verb and noun forms, For this conversion, we use CatVar v2.1 (Habash and Dorr, 2003), which lists categorial variations of words (such as verb *run* for noun *runner*). We also use definitions of the predicates from PropBank (Palmer et al., 2005), which AMR tries to reuse as much as possible, and impose constraints that the defined predicates can only have semantic relations consistent with the definition.

During the training, we use the max-violation perceptron (Huang et al., 2012) with beam size 8 and average the parameters. During the testing, we also perform beam search with beam size 8.

Table 6 shows the overall performance on NP semantic structure analysis. We evaluate the performance using the Smatch score (Cai and Knight,

[2] `https://github.com/jflanigan/jamr`

Method	P	R	F_1
(Flanigan et al., 2014)	75.5	61.1	67.5
Our method (EMPTY/KNOWN)	78.0	63.8	70.2
Our method+LEMMA	75.7	75.2	75.4
Our method+LEMMA/DICT	77.3	77.3	77.3

Table 6: Performance on NP semantic structure analysis

Method	P	R	F_1
(Flanigan et al., 2014)	88.4	71.4	79.0
Our method (EMPTY/KNOWN)	88.9	72.2	79.7
Our method+LEMMA	84.8	84.2	84.5
Our method+LEMMA/DICT	85.8	85.6	85.7

Table 7: Performance on concept identification

2013), which reports precision, recall, and F_1-score for overlaps of nodes, edges, and roots in semantic structure graphs. Compared to the baseline, our method improves both the precision and recall, resulting in an increasing of F_1-score by 2.7 points. When we add the LEMMA rule, the recall increases by 11.4 points because the LEMMA rule can generate concepts that do not appear in the training data, resulting in a further increase of F_1-score by 5.2 points. Finally, when we add the DICT rules, the F_1-score improves further by 1.9 points.

Table 7 shows the performance on concept identification. We report precision, recall, and F_1-score against the correct set of concepts. For each condition, we observe the same tendency in performance increases as Table 6. Thus, we conclude that our method improves both the concept identification and relation identification performances.

5 Conclusion

In this paper, we used Abstract Meaning Representation (AMR) for semantic structure analysis of noun compounds (NPs). We extracted substructures corresponding to NPs from the AMR Bank, and created a data set of NP semantic structures. Then, we proposed a novel method which jointly identifies concepts (nodes) and dependency relations in AMR trees. We confirmed that our method improves the performance on NP semantic structure analysis, and that incorporating an external dictionary further boosts the performance.

Acknowledgements

This work was supported by JSPS KAKENHI Grant Number 15K16053, 26240035.

References

Shu Cai and Kevin Knight. 2013. Smatch: An evaluation metric for semantic feature structures. pages 748–752.

Y.J. Chu and T.H. Liu. 1965. On the shortest arborescence of a directed graph. *Science Sinica*, 14:1396–1400.

Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A. Smith. 2014. A discriminative graph-based parser for the Abstract Meaning Representation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 1426–1436.

Roxana Girju, Dan Moldovan, Marta Tatu, and Daniel Antohe. 2005. On the semantics of noun compounds. *Computer Speech and Language*, 19:479–496.

Nizar Habash and Bonnie Dorr. 2003. A categorial variation database for English. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 17–23.

Jun Hatori, Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2011. Incremental joint POS tagging and dependency parsing in Chinese. In *Proceedings of the 5th International Joint Conference on Natural Language Processing*, pages 1216–1224.

Liang Huang, Suphan Fayong, and Yang Guo. 2012. Structured perceptron with inexact search. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 142–151.

Jacques Janssen and Nikolaos Limnios. 1999. *Semi-Markov models and applications*. Springer, October.

Su Nam Kim and Timothy Baldwin. 2013. A lexical semantic approach to interpreting and bracketing English noun compounds. *Natural Language Engineering*, 19(3):385–407.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pages 423–430.

Maria Lapata. 2002. The disambiguation of nominalizations. *Computational Linguistics*, 28(3):357–388.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP)*, pages 523–530.

Preslav I. Nakov and Marti A. Hearst. 2013. Semantic interpretation of noun compounds using verbal and other paraphrases. *ACM Transactions on Speech and Language Processing*, 10(3):1–51.

Preslav Nakov. 2013. On the interpretation of noun compounds: Syntax, semantics, and entailment. *Natural Language Engineering*, 19(1):291–330.

Joakim Nivre. 2004. Incrementality in deterministic dependency parsing. In *Proceedings of the Workshop on Incremental Parsing: Bringing Engineering and Cognition Together*, pages 50–57.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

Stephen Tratz and Eduard Hovy. 2010. A taxonomy, dataset, and classifier for automatic noun compound interpretation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 678–687.

David Vadas and James R. Curran. 2007. Adding noun phrase structure to the penn treebank. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 240–247.

David Vadas and James R. Curran. 2008. Parsing noun phrase structure with CCG. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 335–343.