# UIT-DANGNT-CLNLP at SemEval-2017 Task 9: Building Scientific Concept Fixing Patterns for Improving CAMR

**Khoa Dang Nguyen**
Faculty of Computer Science
University of Information Technology
VNU-HCM
ndkhoa@nlke-group.net

**Dang Tuan Nguyen**
Faculty of Computer Science
University of Information Technology
VNU-HCM
dangnt@uit.edu.vn

**Abstract**

This paper describes the improvements that we have applied on CAMR baseline parser (Wang et al., 2016) at Task 8 of SemEval-2016. Our objective is to increase the performance of CAMR when parsing sentences from scientific articles, especially articles of biology domain more accurately. To achieve this goal, we built two wrapper layers for CAMR. The first layer, which covers the input data, will normalize, add necessary information to the input sentences to make the input dependency parser and the aligner better handle reference citations, scientific figures, formulas, etc. The second layer, which covers the output data, will modify and standardize output data based on a list of scientific concept fixing patterns. This will help CAMR better handle biological concepts which are not in the training dataset. Finally, after applying our approach, CAMR has scored 0.65 F-score[1] on the test set of Biomedical training data[2] and 0.61 F-score on the official blind test dataset.

## 1 Introduction

Since Abstract Meaning Representation (AMR) was published by Banarescu et al. (2013) for the first time in 2013, it has been considered by many researchers in Natural Language Processing domain. In this trend, the task of AMR has been held continuously for two years in SemEval-2016 and SemEval-2017. There have been many parsers shown its outstanding performance for high F-score points like RIGA (Barzdins and Gosko, 2016), CAMR, CU-NLP (Foland and Martin, 2016), etc.

Inspired by the performance of CAMR in SemEval-2016 Task 8, we selected it as our baseline parser for SemEval-2017 Task 9 - Subtask 1: Parsing Biomedical Data. The parsing task of 2017 has a particular domain but there are many scientific terms, formulas, reference quotations, numbers, etc. That makes the task of 2017 very challenging.

According to Wang et al. (2015a,b), the accuracy of CAMR depends greatly on the accuracy of input dependency parser. When we conduct training and testing on Biomedical training data used for SemEval-2017, we have found that CAMR is not good in handling the reference citations, scientific figures, formulas, etc which are commonly used in scientific papers. This is partly due to the dependency parser not correctly handling this kind of information. And also, the aligner can not fulfill its mission. We have built the first wrapper to support solving the problem related to these information in input data.

At the same time, we found CAMR can memorize very well AMR structure of concepts which have appeared in the training data. However, when parsing testing sentences which have unknown concepts (concepts which are not in training corpus), the parser will not be able to parse and return a single node to indicate the unknown concept (the first error form described in Subsection 3.1). Another weakness of CAMR is the terminal condition. The parser will finish parsing the sentence when the number of elements in the queue has run out. In the output result, we found that many AMR structures of concepts are not in good form (the second error form described in Subsection 3.1). Therefore, we proposed to build a second wrap-

[1]Currently, the datasource for constructing the list of concept fixing patterns are the training, develop and test set of Biomedical training data

[2]This data is all freely available to download at http://amr.isi.edu/download.html

per to fix the parsing error on output data, which will help CAMR deal with these issues better.

## 2 The first wrapper layer

Unlike the corpus used for Task 8 of SemEval-2016, the Biomedical training data corpus used for Task 9 of SemEval-2017 are sentences from scientific articles related to the topic cancer pathway discovery. Generally, the sentences from scientific papers contain a lot of reference citations, scientific figures, formulas, etc.

In the gold AMR structure, the reference citations are represented by node `describe-01`. This node appeared 2,756 times in the training set and the develop set of Biomedical training data. Similarly, in the test set, `describe-01` node appeared 263 times. The number of nodes `describe-01` accounted for a big amount in the corpus. Better handling of these reference citations will increase the accuracy of the parser significantly. However, with the following reference citation formats, input dependency parser is almost impossible to handle and consider the reference citations as meaningless symbols:

1. "[1]"

2. "(1), (2), (3)"

3. "(Wang. et al, 2016)"

Obviously, with the reference citation formats of 1, 2 above, dependency parser will handle these citations as usual numbers. With the reference citation format of 3, it is also not easy for dependency parser because "(Wang. et al, 2016)" does not have the structure of a normal sentence or a clause. Since dependency parser did not handle these quotation formats properly, the aligning and the training process would not achieve good results. For example with the reference citation format of 3, the word "et" can not be aligned to node `other` in the gold AMR. And in the training stage, CAMR won't be able to know that the word "et" is related to node `other`.

And there are also other difficulties such as: the reference citation formats in the corpus have more complex forms than the above examples like "(Wang. et al, 2016a; Wang. et al, 2016b)", there are XML annotations in the input sentences,...

To support solving this problem, we write a tool to remove all XML annotations in the input data,

| Node | Wrapper OFF | Wrapper ON |
|---|---|---|
| describe-01 | 49 | 835 |
| publication-91 | 2 | 497 |
| person | 871 | 993 |
| other | 406 | 501 |
| and | 4731 | 5180 |

Table 1: Number of successfully aligned node before and after the application of the first wrapper

then the tool updates these citation formats to the following patterns:

1. "[1]" → "(described in publication 1)"

2. "(1), (2), (3)" → "(described in publication 1 and 2 and 3)"

3. "(Wang. et al, 2016)" → "(described in publication of Wang and other members in 2016)"

For more complex forms of the reference citation formats, the tool also follows the above patterns. For example, "(Wang. et al, 2016a; Wang. et al, 2016b)" should be updated to "(described in publication of Wang and other members in 2016; described in publication of Wang and other members in 2016)".

These modifications not only help dependency parser can operate more accurately but also help the aligner (Flanigan et al., 2014) align more accurately, especially with node `describe-01` and its sub-nodes (`publication-91`, `person`, `other`, `and`) in the AMR structure.

In addition, we also have a few other modifications with the abbreviations of measuring unit and the abbreviations of scientific term on the first wrapper. The abbreviations of measuring unit can be find out easily by grepping out all `:unit` edges in the traning data. With the abbreviations of scientific term, we have to find them manually. These modifications include:

- Replace the abbreviations of measuring unit with its full form (for example "kDa" → "kilodalton", "pg/ml" → "picogram per milliliter", etc.)

- Replace the abbreviations of scientific term to its full form (for example "UM" → "uveal melanoma")

*Selumetinib in combination with TMZ enhances DNA damage.*

**(a) CAMR parse**

```
(x6 / enhance-01
  :ARG0 (x1 / selumetinib)
  :condition (x3 / combine-01
    :ARG2 (x5 / tmz))
  :ARG1 (x8 / damage-01
    :ARG1 (x7 / enzyme
      :name (n / name
      :op1 "DNA"))))
```

**(b) CAMR parse + our addition**

```
(x6 / enhance-01
  :ARG0 (x1 / small-molecule
    :name (n1 / name
      :op1 "selumetinib"))
  :condition (x3 / combine-01
    :ARG2 (x5 / small-molecule
      :name (n3 / name
        :op1 "TMZ")))
  :ARG1 (x8 / damage-01
    :ARG1 (x7 / nucleic-acid
      :name (n / name
        :op1 "DNA")
      :wiki "DNA")))
```

Figure 1: An example of CAMR parsing result and our addition

Table 1 represents the number of aligned nodes before and after the application of our first wrapper. Obviously the number of aligned nodes is increased substantially. Especially with nodes often appear in AMR structure describe the reference citations. The more successfully aligned nodes, the easier for CAMR in the traning stage.

## 3 The second wrapper layer

### 3.1 Parsing error detecting method

We classify parsing errors of CAMR into two main types. First, we propose methods to identify all the errors of the two types of the returned output of CAMR. Figure 1 represents the results of CAMR's original parsing result and the parsing result after being updated by our system.

The first error type usually happens with the unknown concept, which does not appear in the training set. When processing an unknown concept, CAMR will return an individual node such as `(x1 / selumetinib)` or `(x5 / tmz)`. To be able to identify the errors of this type in the returned results, we will collect a list of all of node labels in gold AMR of training set. From this list, we will traverse through all nodes on AMR results, if any node label is not on this list, it's likely that node is a presentation for an unknown concept. For example, with the AMR in Figure 1b, then this list is: `{enhance-01, small-molecule, name, "selumetinib", combine-01, "TMZ", damage-01 , nucleic-acid, "DNA"}`. Node `selumetinib` and `tmz` (without quotes) are not in the list above, so these may be unknown concepts.

Second error form is related to the structure of the AMR node such as wrong concept type, missing important sub-node or having wrong sub-node, null node, null edge, etc. As in Figure 1a, the `"DNA"` node is identified as an `enzyme`. But in the training set, there is no `enzyme` node which has the `:name` edge connected to a `"DNA"` node. `"DNA"` is always a `nucleic-acid` in the training set. The method to identify this error type is the same as above, but instead just collect a list of node labels, we will collect a list of all concept nodes in the training set. A node is called concept node when it has a direct `:name` edge. We call this list is the list of the AMR structure concept. When traversing through all nodes in the result AMR, if there is a `:name` edge appeared in a node of the AMR structure, we will compare that node with these nodes in the list of the AMR structure concept. If the list doesn't contain that node, that mean this is a new node created by CAMR. And there is a probability that the node contains some errors. We just stop at identifying concept node in AMR structure which has the probability of containing errors. It still needs the help of a human expert to give out a final judgment that the concept node of AMR structure is correct or not.

### 3.2 Parsing error fixing method

From the two error lists above, we will build a new list with each element has the form "Label-Error AMR-Fixed AMR". We call this the list of concept fixing patterns. Two ingredients Label and Error AMR are taken directly from the two error lists above. Filling the Fixed AMR would require the support of human experts. Particularly for Biomedical training data, we refer the gold AMR in the test set of training data to fill out the Fixed AMR.

Table 2 represents the list of concept fixing patterns for the example in Figure 1.

After having the parsing result of CAMR as in Figure 1a. We will traverse all the

| Label | selumetinib |
|---|---|
| Error AMR | `(x1 / selumetinib)` |
| Fixed AMR | `(x1 / small-molecule :name (n1 / name :op1 "selumetinib"))` |
| Label | tmz |
| Error AMR | `(x5 / tmz)` |
| Fixed AMR | `(x5 / small-molecule :name (n3 / name :op1 "TMZ"))` |
| Label | enzyme |
| Error AMR | `(x7 / enzyme :name (n / name :op1 "DNA"))` |
| Fixed AMR | `(x7 / nucleic-acid :name (n / name :op1 "DNA") :wiki "DNA")` |

Table 2: The list of concept fixing patterns for the example in Figure 1

nodes in the result AMR. When traversing node `(x1 / selumetinib)`, we will find out these elements in the list of concept fixing patterns which have Label is `selumetinib`. We will compare the structure of traversing node with the structure in Error AMR part of these elements. If there are a structural matching, we will replace the traversing node structure with the structure in the Fixed AMR part of the corresponding element. Similarly, the structure `(x5 / tmz)` and `(x7 / enzyme :name (n / name :op1 "DNA"))` will also be updated to the Fixed AMR structure in the list of concept fixing patterns. The final output will the the same as the AMR structure in 1b.

## 4 Experiment

We used the Biomedical training data which have been split into training, develop and test set for experiments. About CAMR, we used the version which was described in (Wang et al., 2016) as a baseline parser with its default configurations. But, we have not used named entity tags and semantic role labels in the experiment stage. To evaluate the output result, we used the Smatch tool (Cai and Knight, 2013) at version 16.11.14.

Firstly, we implemented the first wrapper layer as the proposed method in Section 2. Then, we started training on two systems. The training data is the collection of all sentences in training set and develop set of Biomedical training data. On the first system, CAMR was trained with the original training data, which had not been updated by our first wrapper layer. On the other system, the training data had been updated by our first wrapper layer. Both of two systems were trained in 10 iterations.

| System | Precision | Recall | F-score |
|---|---|---|---|
| Baseline parser | 0.67 | 0.50 | 0.57 |
| OurSystem(W1) | 0.70 | 0.53 | 0.60 |
| OurSystem(W12) | 0.73 | 0.58 | 0.65 |

Table 3: Comparison with the baseline parser

After that, two systems were tested with all sentences from the test set of Biomedical training data. In order to fix the parsing error of CAMR, we need to build the list of concept fixing patterns. We implemented a tool to detect the concept parsing errors. This tool will traverse all the nodes in the output result of CAMR when parsing test set of training data to collect a list of AMR node which has the probability of containing errors as the proposed method in Subsection 3.1. After finished detecting error, the tool will return the list of concept fixing patterns. Each element of the list will have form of "Label-Error AMR-Fixed AMR". We then refer the gold AMR in the training data to fill out the "Fixed AMR" part. We have to do this manually to guarantee that the detected error node actually contains error.

After having the list of concept fixing patterns, we will have another tool to update the parsing result based on the list. We only run this tool on the second system. The output result of the first system is keep original. Table 3 shows our experiment results. The first row is the evaluated score of the baseline parser. The second row is the score of our system when only used the first wrapper layer. The last row is the score of our system when have both two wrappers activated. There is a special note about wrapper 2: currently, we have to extract the data from the training, develop and test set of the Biology training data to fill out the "Fixed AMR" part.

With the official blind test set, our system used the first wrapper layer to normalize and update input sentences. Then, our system automatically used the above pre-built list of concept fixing patterns to fix the parsing error of CAMR before return the final output result. We achieved 0.61 F-score on the official blind test set.

## 5 Conclusion

The main contribution of this paper is the second wrapper layer which can be used very effective in finding concept parsing errors of AMR parsers. Although the improvements have been developed in CAMR, but both of these two wrappers can easily be applied to other AMR parsers.

However, currently, our approach requires manual processing to create the concept fixing patterns. In further work, we will focus on researching about automatically create the fixing patterns for a few particular domains. We intend to use open knowledge databases of these domains, the combination with supervised machine learning methods and pre-built corpus to create the concept fixing patterns automatically.

## References

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*. Association for Computational Linguistics, pages 178–186. http://www.aclweb.org/anthology/W13-2322.

Guntis Barzdins and Didzis Gosko. 2016. Riga at semeval-2016 task 8: Impact of smatch extensions and character-level neural translation on amr parsing accuracy. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*. Association for Computational Linguistics, pages 1143–1147. https://doi.org/10.18653/v1/S16-1176.

Shu Cai and Kevin Knight. 2013. Smatch: an evaluation metric for semantic feature structures. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, pages 748–752. http://aclweb.org/anthology/P13-2131.

Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and A. Noah 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 (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1426–1436. https://doi.org/10.3115/v1/P14-1134.

William Foland and H. James Martin. 2016. Cu-nlp at semeval-2016 task 8: Amr parsing using lstm-based recurrent neural networks. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*. Association for Computational Linguistics, pages 1197–1201. https://doi.org/10.18653/v1/S16-1185.

Chuan Wang, Sameer Pradhan, Xiaoman Pan, Heng Ji, and Nianwen Xue. 2016. Camr at semeval-2016 task 8: An extended transition-based amr parser. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*. Association for Computational Linguistics, San Diego, California, pages 1173–1178. http://www.aclweb.org/anthology/S16-1181.

Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015a. Boosting transition-based amr parsing with refined actions and auxiliary analyzers. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Association for Computational Linguistics, Beijing, China, pages 857–862. http://www.aclweb.org/anthology/P15-2141.

Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015b. A transition-based algorithm for amr parsing. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Denver, Colorado, pages 366–375. http://www.aclweb.org/anthology/N15-1040.