DANGNT@UIT.VNU-HCM at SemEval 2019 Task 1: Graph Transformation System from Stanford Basic Dependencies to Universal Conceptual Cognitive Annotation (UCCA)

Dang Tuan Nguyen and Trung Tran
University of Information Technology, VNU-HCM
Ho Chi Minh City, Vietnam
dangnt@uit.edu.vn, ttrung@nlke-group.net

Abstract

This paper describes the graph transformation system (GT System) for SemEval 2019 Task 1: Cross-lingual Semantic Parsing with Universal Conceptual Cognitive Annotation (UCCA)[1]. The input of GT System is a pair of text and its unannotated xml, which is a layer 0 part of UCCA form. The output of GT System is the corresponding full UCCA xml. Based on the idea of graph illustration and transformation, we perform four main tasks when building GT System. At the first task, we illustrate the graph form of stanford dependencies[2] of input text. We then transform into an intermediate graph in the second task. At the third task, we continue to transform into ouput graph form. Finally, we create the output UCCA xml.

The evaluation results show that our method generates good-quality UCCA xml and has a meaningful contribution to the semantic representation sub-field in Natural Language Processing.

1 Introduction

In the past few years, semantic representation is receiving growing attention in NLP. Researchers have recently proposed different semantic schemes. Examples include Abstract Meaning Representation (Banarescu et al. 2013), Broad-coverage Semantic Dependencies (Oepen et al. 2014), Universal Decompositional Semantics (White et al. 2016), Parallel Meaning Bank (Abzianidze et al. 2016), Universal Conceptual Cognitive Annotation (Abend and Rappoport 2013). These advances in semantic representation, along with corresponding advances in semantic parsing, text understanding, summarization, paraphrase detection, and semantic evaluation.

In SemEval 2019 Task 1: Cross-lingual Semantic Parsing with Universal Conceptual Cognitive Annotation (UCCA)[1], the Committee focuses on parsing text according to the UCCA semantic annotation. UCCA (Abend and Rappoport 2013) is a cross-linguistically applicable semantic representation scheme, based on Basic Linguistic Theory (Dixon 2010). In general, UCCA represents the semantics of linguistic utterances as directed acyclic graphs (DAGs). In one DAG, nodes and edges belong to one of several layers. There are two types of node: (i) terminal nodes express the text tokens; (ii) non-terminal nodes express semantic units. Edges are labelled, indicating the role of a child in the relation the parent represents. As an example, consider sentence in Example 1: "*The album was recorded in Switzerland .*". Two layers of UCCA xml of this sentence:

- Layer0:

```
<root annotationID="0" passageID="503012">
     <attributes />
     <layer layerID="0">
       <attributes />
       <extra ... />
       <node ID="0.1" type="Word">
         <attributes ... text="The" />
         <extra dep="det" ... tag="DT" />
       </node>
       ...
     </layer>
</root>
```

The relations of NodeID and corresponding lexicon:

```
{[ID="0.1" ➔ dep="det" ➔ "The"]
[ID="0.2" ➔ dep="nsubj:pass" ➔ "album"]
[ID="0.3" ➔ dep="aux:pass" ➔ "was"]
[ID="0.4" ➔ dep="root" ➔ "recorded"]
[ID="0.5" ➔ dep="case" ➔ "in"]
[ID="0.6" ➔ dep="obl" ➔ "Switzerland"]
[ID="0.7" ➔ dep="punct" ➔ "."]}
```

- Layer1:

```
<layer layerID="1">
       <attributes />
       <node ID="1.1" type="FN">
         <attributes />
```

[1] https://competitions.codalab.org/competitions/19160

Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019), pages 97–101
Minneapolis, Minnesota, USA, June 6–7, 2019.

```
      <edge toID="1.2" type="H">
        <attributes />
      </edge>
    </node>
    <node ID="1.2" type="FN">
      <attributes />
      <edge toID="1.4" type="A">
        <attributes />
      </edge>
      ...
    </node>
    <node ID="1.4" type="FN">
      <attributes />
      <edge toID="1.10" type="E">
        <attributes />
      </edge>
      ...
    </node>
    <node ID="1.10" type="FN">
      <attributes />
      <edge toID="0.1" type="Terminal">
        <attributes />
      </edge>
    </node>
    ...
  </layer>
```

We have the graphical representation of the above UCCA:

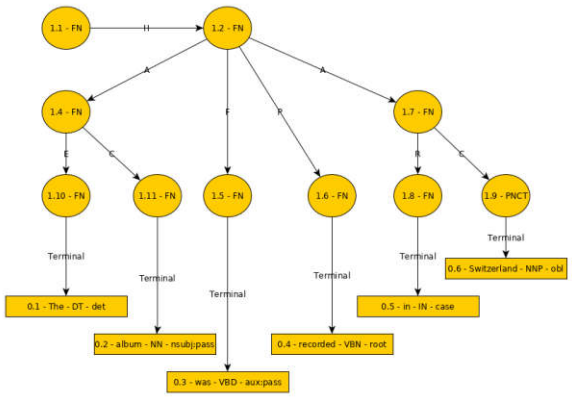

Figure 1: Graph form of UCCA xml of sentence in Example 1.

The primary purpose of this article is to present our system called graph transformation system (GT System) for Task[1]. We perform four tasks when building GT System. At the first task, we illustrate the graph form of Stanford dependencies[2] (Manning et al. 2014; Marie-Catherine et al. 2014) of input text. We then transform into an intermediate graph in the second task. At the third task, we continue to transform into ouput graph form. Finally, we create the output UCCA xml.

The rest of article is separated as follows. We briefly describe Stanford dependencies in Section 2. In Section 3, we introduce our GT system for Task[1]. Section 4 details the experiments and analyzes the results. We offer conclusions in Section 5.

2 Stanford Dependencies

Stanford dependencies[2] (Manning et al. 2014; Marie-Catherine et al. 2014; Marie-Catherine and Manning 2008) provides a representation of grammatical relations between words in a sentence. Stanford dependencies (SD) have three parts: name of the relation, governor and dependent. Consider English sentence in Example 1, below is the xml representation of SD basic dependencies. This representation is the result of running Stanford CoreNLP pipeline (Manning et al. 2014).

```
<dependencies type="basic-dependencies">
    <dep type="root">
      <governor idx="0">ROOT</governor>
      <dependent idx="4">recorded</dependent>
    </dep>
    <dep type="det">
      <governor idx="2">album</governor>
      <dependent idx="1">The</dependent>
    </dep>
    <dep type="nsubjpass">
      <governor idx="4">recorded</governor>
      <dependent idx="2">album</dependent>
    </dep>
    <dep type="auxpass">
      <governor idx="4">recorded</governor>
      <dependent idx="3">was</dependent>
    </dep>
    <dep type="case">
      <governor
idx="6">Switzerland</governor>
      <dependent idx="5">in</dependent>
    </dep>
    <dep type="nmod">
      <governor idx="4">recorded</governor>
      <dependent
idx="6">Switzerland</dependent>
    </dep>
</dependencies>
```

We have the graphical representation of the above SD basic dependencies:

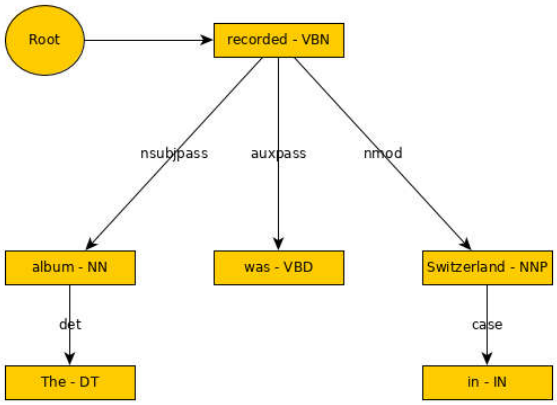

Figure 2: Graph form of SD basic dependencies of sentence in Example 1.

[2] https://stanfordnlp.github.io/CoreNLP/

3 The Graph Transformation System

In this section, we express our GT system for creating UCCA xml of the input text. The general architecture is represented in Figure 2:

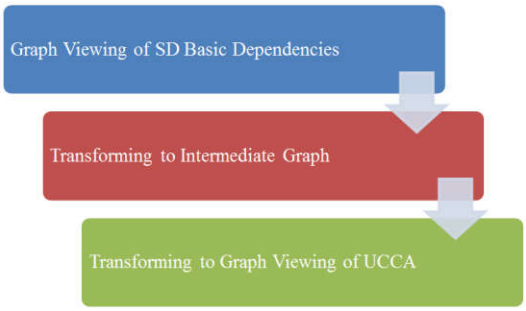

Figure 3: Architecture of Graph Transformation System.

When building GT System, we perform two processes: training and testing process. At training process, we build the intermediate graph from UCCA and SD basic dependencies of training data[1]. At the testing process, which can be called the inverse process of training, we build the ouput UCCA from intermediate graph of testing data.

3.1 Intermediate Graph

In general, the intermediate graph is an irreducible representation of UCCA graph form. This intermediate graph is quite similar to graph form of SD basic dependencies. The main difference of the intermediate graph and graph form of SD basic dependencies is: each edge label in the intermediate graph is the combination of UCCA categories (Abend and Rappoport. 2013) and Stanford dependency relations (Marie-Catherine and Manning 2008a, 2008b).

Below is the intermediate graph of sentence in Example 1. This graph is the reduction of graph in Figure 1, and quite similar to graph in Figure 2.

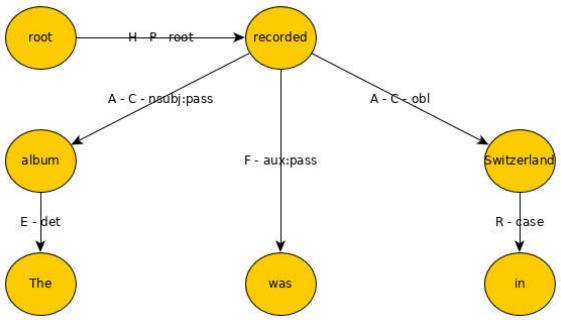

Figure 4: Intermediate graph of sentence in Example 1.

3.2 Training Process

Firstly, at training process, we consider train data[1] and performed main tasks. The first and second task is in turn viewing the graph from of SD basic dependencies and UCCA of input text. At the third task, we propose Left-First-Search liked algorithm with Bottom-Up idea to reduce the graph form of UCCA to intermediate graph. At the final task, we propose rules and heuristics for matching graph form of SD basic dependencies and intermediate graph.

The main steps of Left-First-Search (LFS) algorithm is as follow. ***Step 1***. Browse to terminal on the left. ***Step 2***. Back to parent node of this terminal. Check if parent having any other child or not. Step 2.1. If yes. Repeat Step 1 with root is this child node. ***Step 3***. Swap the position of root of sub-tree with position of child having important annotation. ***Step 4***. Back to parent node of this root. Repeat Step 2 with this parent.

To perform LFS algorithm, we determine the priority of SD and UCCA annotations according to two factors. **First**. The meaning of each annotation, representing the dependency relations and grammatical roles of lexicons. **Second**. The position of each node in graph.

Apply LFS algorithm for graph in Figure 3, we in turn have three level reductions in Figure 5, 6, 4 (respectively):

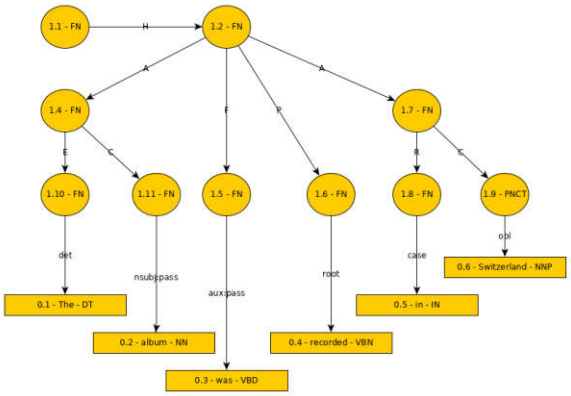

Figure 5: First reduction of Graph form in Figure 1.

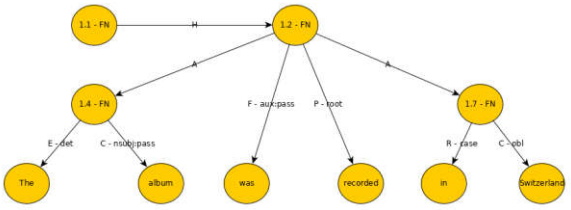

Figure 6: Second reduction of Graph form in Figure 1.

After having the final reduction, which is intermediate graph, of graph form of UCCA, we compare with graph form of SD basic dependencies. We consider the similarities between two graphs and propose rules and heuristics to (i) determine the level of one node, and (ii) determine the group

of UCCA annotation for each level. The general idea of mechanism is:

- Collect all SD-type of relations in UCCA and SD basic dependencies of training data. Below is the collection:

SD basic dependencies	acl:relcl / expl / csubjpass / cop / aux / conj / acl / xcomp / dep / appos / advmod / neg / det / cc:preconj / nmod:tmod / ccomp / root / advcl / nsubj / case / iobj / cc / det:predet / nmod:poss / compound:prt / csubj / nsubjpass / nummod / nmod:npmod / nmod / auxpass / parataxis / amod / compound / discourse / mwe / dobj / mark
UCCA	acl:relcl / expl / obl:npmod / cop / aux / conj / acl / appos / xcomp / goeswith / advmod / det / ccomp / nsubj:pass / cc:preconj / nmod:tmod / flat / root / obl:tmod / advcl / punct / nsubj / case / iobj / cc / vocative / det:predet / nmod:poss / compound:prt / csubj / nummod / nmod:npmod / nmod / parataxis / amod / list / compound / discourse / aux:pass / obj / obl / fixed / mark

- Determine the priority order of SD-type relations.

Example 2: dobj -> amod -> dep -> nmod -> case.

- Determine the compound (UCCA and SD) relation in each node level.

Example 3: type conj at level 7: "H - A - E - C - C - C - conj"

3.3 Testing Process

At the testing process, which can be called the inverse process of training, we considered development and test data[1] and performed main tasks. The first task is viewing the graph from of SD basic dependencies of input text. At the second task, we applied proposed rules and heuristics to transform this graph to intermediate graph. We then, at the final task, we proposed Breadth-First-Search liked algorithm with Top-Down idea to re-create the graph form of UCCA from intermediate graph. This BFS algorithm is, in fact, the inverse mechanism of LFS algorithm in Section 3.2.

The main steps of Breadth-First-Search (BFS) algorithm is as follow. ***Step 1***. Reduce the first level of node. ***Step 2***. Determine the intergratedChild which adheres to this node. ***Step 3***. If there is no intergratedChild. Step 3.1. Repeat Step 1 until node come down to terminal position. Step 3.2. Repeat from Step 1 to Step 4 with each child of this node. ***Step 4***. If there is intergratedChild. Step 4.1. Repeat from Step 1 to Step 4 with each child of this node which are different from intergratedChild. Step 4.2. Repeat from Step 1 to Step 4 with this node. Step 4.3. Repeat from Step 1 to Step 4 with intergratedChild.

4 Experiment and Evaluation

At the evaluation phase, we focus on English in-domain setting, using the Wiki corpus. In testing data, this domain consists of 515 small texts with corresponding unannotated UCCA xmls.

We test our method for both open and closed track in the English setting: (i) closed track submission is only allowed to use the gold-standard UCCA annotation distributed for the task in the target language, and limited in its use of additional resources; (ii) open track submission is allowed to use any additional resource.

Table 1 and 2 view the results of testing data for open and closed tracks with labeled (first row) and unlabeled scores (second row).

Averaged F1		**P**	**R**	**F1**
0.708	Primary	0.738	0.694	0.715
	Remote	1.000	0.000	0.000
0.822	Primary	0.857	0.806	0.831
	Remote	1.000	0.000	0.000

Table 1: Results of testing data in open track.

Averaged F1		**P**	**R**	**F1**
0.706	Primary	0.737	0.692	0.714
	Remote	1.000	0.000	0.000
0.825	Primary	0.860	0.808	0.833
	Remote	1.000	0.000	0.000

Table 2: Results of testing data in closed track.

The testing results show that our GT system creates good quality UCCA semantic representations in English Wiki testing data.

5 Conclusion

We have presented the graph transformation method for creating UCCA semantic representation from English in-domain setting, using the Wiki corpus[1]. Our method performs four main tasks: (i) illustrate the graph form of Stanford dependencies[2] of input text; (ii) transform into an intermediate graph; (iii) continue to transform into ouput graph form; (iv) create the output UCCA xml. The experiment results show that our method meets the requirements from SemEval Task[1].

In future works, we intend to improve the transformational algorithms and propose more accurate rules for selecting best nodes and dependency tags. Besides, we expand our method and test with other datasets for a broader comparison.

References

Aaron Steven White, Drew Reisinger, Keisuke Sakaguchi, Tim Vieira, Sheng Zhang, Rachel Rudinger, Kyle Rawlins, Benjamin Van Durme. 2016. Universal Decompositional Semantics on Universal Dependencies. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas, pages 1713–1723.

Daniel Hershcovich, Omri Abend and Ari Rappoport. 2017. A Transition-Based Directed Acyclic Graph Parser for UCCA. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Vancouver, Canada, pages 1127–1138.

Daniel Hershcovich, Omri Abend and Ari Rappoport. 2018. Multitask Parsing Across Semantic Representations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Long Papers)*. Association for Computational Linguistics, Melbourne, Australia, pages 373–385.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal Dependencies v1: A Multilingual Treebank Collection. In *LREC 2016*.

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.

Lasha Abzianidze, Johannes Bjerva, Kilian Evang, Hessel Haagsma, Rik van Noord, Pierre Ludmann, Duc-Duy Nguyen, Johan Bos. 2017. The Parallel Meaning Bank: Towards a Multilingual Corpus of Translations Annotated with Compositional Meaning Representations. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*. Valencia, Spain, pages 242–247.

Manning, Christopher D., Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55-60.

Marie-Catherine de Marneffe, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre, and Christopher D. Manning. 2014. Universal Stanford Dependencies: A cross-linguistic typology. In *Proceedings of LREC*.

Marie-Catherine de Marneffe and Christopher D. Manning. 2008a. The Stanford typed dependencies representation. In *COLING 2008 Workshop on Cross-framework and Cross-domain Parser Evaluation*.

Marie-Catherine de Marneffe and Christopher D. Manning. 2008b. *Stanford Dependencies manual*.

Omri Abend and Ari Rappoport. 2013. Universal Conceptual Cognitive Annotation (UCCA). In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Sofia, Bulgaria, pages 228–238.

Omri Abend, Shai Yerushlami and Ari Rappoport. 2017. UCCAApp: Web-application for Syntactic and Semantic Phrase-based Annotation. In *Proceedings of ACL 2017*.

Robert M. W. Dixon. 2010. *Basic Linguistic Theory: Grammatical Topics, Volume 2*. Oxford University Press.

Sebastian Schuster and Christopher D. Manning. 2016. Enhanced English Universal Dependencies: An Improved Representation for Natural Language Understanding Tasks. In *LREC 2016*.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Dan Flickinger, Jan Hajic, Angelina Ivanova, and Yi Zhang. 2014. SemEval 2014 Task 8:Broad-Coverage Semantic Dependency Parsing. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*. Dublin, Ireland, pages 63–72.