UMR-Writer 2.0: Incorporating a New Keyboard Interface and Workflow into UMR-Writer

Sijia Ge [1*], **Jin Zhao** [2*], **Kristin Wright-Bettner**[1], **Skatje Myers**[1]
Nianwen Xue[2], **Martha Palmer**[1]
[1] University of Colorado at Boulder
[2] Brandeis University
{sijia.ge,kristin.wrightbettner,skatje.myers,
martha.palmer}@colorado.edu
{jinzhao,xuen}@brandeis.edu

Abstract

UMR-Writer is a web-based tool for annotating semantic graphs for the Uniform Meaning Representation (UMR) scheme. UMR is a graph-based semantic representation that can be applied cross-linguistically for deep semantic analysis of text. In this work, we implemented a new keyboard interface for UMR-Writer 2.0, which adds to the original click-based interface to support faster annotation for more experienced annotators. The new interface also addresses some issues with the original click-based interface. Additionally, we demonstrate an efficient workflow for annotation project management in UMR-Writer 2.0, which has been applied to many projects.

1 Introduction

UMR-Writer (Zhao et al., 2021) is a web-based application used for annotating Uniform Meaning Representation (UMR). UMR is a graph-based, cross-linguistically applicable semantic representation designed to support interpretable natural language applications that require deep semantic analysis (Gysel et al., 2021; Bonn et al., 2023). It captures the meaning of natural language sentences and documents in a structured, human- and machine-readable format (Figure A1 shows a complete UMR graph).

UMR is an extension of Abstract Meaning Representation (AMR, Banarescu et al., 2013) and enriches the AMR semantic scheme to cover additional linguistic categories such as aspect (Donatelli et al., 2018; Van Gysel et al., 2019), and scope (Pustejovsky et al., 2019) in sentence-level annotation. UMR also supports document-level annotation for temporal relations (Yao et al., 2020), modality (Vigus et al., 2019), and coreference (O'Gorman et al., 2018). Moreover, UMR is also a universal multi-language semantic scheme that can be used to annotate low-resource languages such as Arapaho, Kukama, and Secoya, etc (Van Gysel et al., 2021; Vigus et al., 2020).

As graphs, UMR can be serialized into triples (parent concept node, relation, child concept node). Parent and child nodes can be abstract concepts, lexicalized concepts, or attribute values. Relations can be roles or other types of semantic relations. UMR-Writer originally has a click-based interface for annotators to construct UMR triples at the sentence level. An example is shown in Figure 1, which requires five steps for annotating the concept "*free*" in the sentence "*Edmund Pope tasted freedom today for the first time in more than eight months*". Annotators could 1) select a parent concept node "*taste*" by clicking the node; 2) select the child concept node "*free*" by selecting a span from the raw text; 3) look up senses by clicking the "lexicalized concept" box. Then, by hovering the cursor, annotators could 4) view the frame information and choose the correct concept sense, and finally 5) choose the correct relation (here, "*ARG1*", proto-patient) from the corresponding drop-down menus.

This approach creates several issues during annotation. Firstly, many annotators have extensive experience in annotating AMR with the AMR editor (Hermjakob, 2013). It uses a keyboard interface to annotate AMR graphs by entering editing commands. Therefore, annotators who are accustomed to the AMR editor may prefer to keep the keyboard interface instead of learning how to annotate in a click-based interface from scratch. Secondly, the multiple complicated drop-down menus in the click-based interface often confuse and overwhelm annotators. Annotators need to move the mouse back and forth between multiple drop-down menus and the sentence itself in order to add just one node to the graph, in addition to simultaneously paying attention to the sentence-level UMR graph, as shown in Figure 1. This impacts the annotation

[*] These authors contributed equally to this work.

Proceedings of the 17th Linguistic Annotation Workshop (LAW-XVII), pages 211–219

efficiency and quality. Finally, some concepts are non-sequential in some languages, and it is tricky to select multiple non-sequential spans with a mouse.

To address these issues, we implemented a keyboard interface in UMR-Writer 2.0 (§3). The new interface was developed using Flask, JavaScript/Jquery, HTML/CSS, and PostgreSQL. It is specifically designed for sentence-level annotation and coexists with the original click-based interface, allowing annotators to choose their preferred approach. The interface for document-level annotation remains unchanged. Besides the annotation procedure, in terms of the workflow set-up, users also reported that managing annotation data for multiple corpora becomes difficult with the increasing size of the annotation. Thus this paper also introduces an efficient workflow for project management (§4), and other features for UMR-Writer 2.0 [1].

2 Related Tools

AMR editor is an easily accessible web-based annotation tool for AMR with comprehensive functionalities (Hermjakob, 2013). It is a command-based tool where annotators can enter short editing commands to annotate AMR graphs. Besides the basic function of building AMR graphs, it offers many useful features such as copy and paste of partial graphs, searching, and administrative support. Many features of the keyboard interface in this paper are inspired by the AMR editor. However, the AMR editor does not support document-level annotation and languages other than English.

There are other annotation tools available, such as Anafora (Chen and Styler, 2013) and BRAT (Stenetorp et al., 2012). Anafora is a web-based text annotation tool that is lightweight, flexible, easy to use, and capable of annotating with a variety of schemas. BRAT offers visualization for annotators to intuitively figure out the relations across text annotations. However, neither of these annotation tools is compatible with the UMR scheme and annotation requirements because they cannot annotate the concepts in the form of word lemmas, concatenated words, or abstract concepts that do not correspond to any specific word tokens in the source text. Like Anafora and BRAT, UMR-Writer can be modified to extend its usage to other graph-based formalisms besides UMR in theory, making it a versatile annotation tool. These modifications include customizing the relations and concept types to meet the requirements of various annotation tasks.

3 The Keyboard Interface of UMR-Writer 2.0

We first overview the layout of the new keyboard interface, then introduce the annotation methods and related functionalities.

3.1 Layout

Compared with the original click-based interface of UMR-Writer shown in Figure 1, the keyboard interface removes the drop-down menus on the right since annotators no longer need to interact with them. Instead, annotators enter the editing commands. To input commands, we added an input box under the raw text.

In the click-based interface, there is insufficient space to directly display the frame information, requiring annotators to hover the cursor over the predicate's sense to view the frame. In the new keyboard interface, we leverage the space created by removing the drop-down menus to display the frame information directly to annotators. The overall layout is shown in Figure 2. Annotators can primarily focus on the left-most area of the interface, which includes the raw text, editing command, and the generated UMR graph. This reduces the need for excessive eye and mouse movements associated with the click-based interface.

3.2 UMR Input Methods

To construct UMR graphs, we adopt the same "typing" method as the AMR editor for annotation but use an index-based style command (Li et al., 2016). The tool assigns a "superscript" to each token to signify its 1-based indexing position in the raw text. Annotators add an "x" before the index to refer to the token in the raw text. For example:

```
Edmund[1] Pope[2] tasted[3] freedom[4]
today[5] for[6] the[7] first[8] time[9] in[10]
more[11] than[12] eight[13] months[14]
```

In this example, the first token "*Edmund*" is "x1", the second token "*Pope*" is "x2", and so on. The tool keeps track of tokens entered by the annotator and queries the lemmas from the database to obtain the corresponding concepts. It then displays the corresponding PropBank-style frame (Palmer

[1] UMR-Writer can access via the link: http://umr-tool.cs.brandeis.edu/.

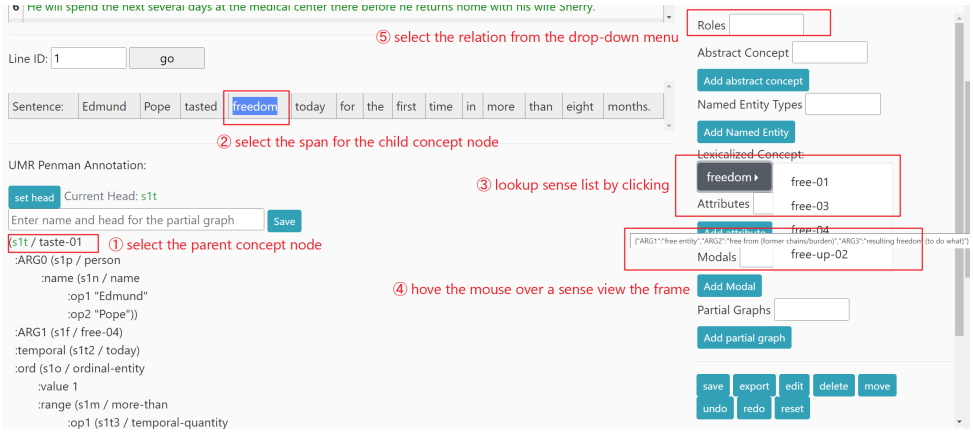

Figure 1: The click-based interface

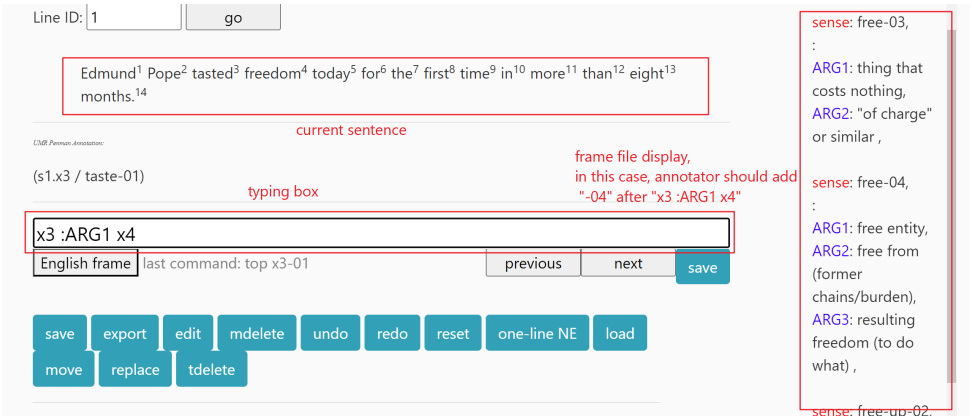

Figure 2: The keyboard interface

et al., 2005; Pradhan et al., 2022) information in the area to the right of the annotator. If annotators need to choose the correct sense from the current predicate's frame, they only have to attach the sense number with a dash marker following the index, such as "`x3-01`". This represents the first sense of the concept "*taste*" and indicates that it is the third token in the raw text. Annotators can input commands such as "`x3 :ARG1 x4-04`" shown in Figure 2 for annotating concept "*free*". This represents the fourth sense of the concept, which is the fourth token "*freedom*" in the text, acting as the "*ARG1*" (proto-patient) of its parent node, the concept "*taste*" (the sense number only needs to be specified once). The tool will then add a node to the UMR graph. When annotating abstract concepts such as named entities, annotators can enter a command such as "`x3 :ARG0 person x1_x2`" by attaching the abstract concept label before the index.

Additionally, such an approach using index-based command is applicable to situations where a concept is composed of multiple tokens or parts of a token resulting from segmentation errors or other reasons. In particular, it addresses the issue that a concept may consist of several non-sequential tokens such as the phenomenon of "Ionization of Pseudo-V-O Compounds" in Chinese (Chao, 1968), e.g.,[2]

(1) 我[1] 先[2] 给[3] 你[4] 提[5] 个[6] 醒[7]
1SG first give 2SG **warn** CLS **reminder**

'I'll first give you a reminder.'

In this case, the fifth token and the seventh token should be considered as a whole compound "提醒" (make aware), but other grammatical elements, such as the noun classifier, can be freely inserted into the middle, making the word look like a V-O construction, even though it makes no sense to interpret the two tokens separately. It can be tricky to select multiple non-sequential spans with a mouse, but annotators can enter commands like "`x5_x7`" to represent the concept consisting of the fifth and the seventh tokens.

[2] 1SG = first person singular, 2SG = second person singular, CLS = noun classifier.

The tool does not adopt the method used in the AMR editor that requires typing the concept directly for three reasons:

1. It can be difficult to record the alignment information, which is crucial for UMR annotation. Explicitly representing the correspondence between word tokens in the sentence and the concepts/relations in the UMR graph is useful for automatic parsing.
2. Entering a concept creates a higher probability of accidental typos compared with entering an index.
3. Annotators may need to frequently switch input method editors (IME) for languages such as Chinese and Arabic that are not based on an alphabet writing system to input commands.

Along with input commands, we also change variables represented in PENMAN (Kasper, 1989; Goodman, 2020) notation for concepts in UMR graphs. Each concept is associated with a variable that uniquely identifies a graph node. Variables serve as "shorthand" references for concepts, for example:

t / taste-01

"t" is the variable represented in PENMAN notation for the concept "*taste*". It uses the initials of the concept and a ascending number to distinguish between concept nodes with the same initial in AMR. The click-based interface follows the same convention but adds a sentence number to form strings such as "s2t" for document-level annotation ("s2" represents the second sentence).

In the keyboard interface, we concatenate the index after the auto-generated sentence number to form variables such as "s2x3" instead of taking the initials of concepts. This is because using the initials as variables is not feasible for languages that do not have an alphabet-based writing system. Additionally, initials-based variables cannot differentiate abstract concepts from lexicalized concepts. In the keyboard interface, each abstract concept is assigned a variable with an index that exceeds the total number of tokens in the sentence, and it is marked as an abstract concept with the prefix "ac" instead of the prefix "x" used for lexicalized concepts. For example, in the previous sentence "*Edmund Pope tasted freedom today for the first time in more than eight months*", the phrase "*eight months*" corresponds to an abstract concept "*temporal-quantity*", and since the number of tokens in this example is 14, we can assign the variable "s2ac15" to the "*temporal-quantity*" concept. Moreover, while the index of a token in a text is fixed, initial-based variables can vary based on the annotation order for concepts with the same initials. If we use the index as the input command to annotate a concept and then later on adopt initial-based variables, it would result in inconsistency. The index-based variables also encode alignment information.

The above changes in the keyboard interface make the annotation process more efficient, reducing five steps required in the click-based interface (Figure 1) to a single command (Figure 2) for adding a node in UMR graphs.

3.3 Other Functionalities for Editing UMR Graphs

We have implemented additional functionalities that go beyond adding a single concept node. For example, annotators can edit UMR graphs and they can delete an incorrect partial graph by clicking on its parent concept node. This action will delete both the parent node and its descendant nodes. Annotators can also move a partial graph to a different location instead of deleting and recreating it. In addition, annotators can use the new "redo" and "undo" buttons to recover from mistakes and track their editing progress. Furthermore, they can name and save partial graphs for future use, or copy-and-paste a partial graph directly from another annotation when constructing a new graph.

Overall, these additional functionalities enhance efficiency and flexibility, making the annotation process more convenient and effective.

4 Annotation as Projects

The annotation process can become messy and disorganized if an annotator works on multiple corpora. To address this issue, we have introduced the "project" concept in UMR-Writer 2.0.

Each project folder contains two sub-folders for storing completed annotations submitted by annotators: The first sub-folder is called "Quality Control" (QC), which stores the final version of each annotation file, and the second sub-folder is called "Double Annotated" (DA), which preserves multiple copies of the same file annotated by different annotators.

To manage the annotation projects effectively, we have created an administrative permission hierarchy. The hierarchy of administrative permissions

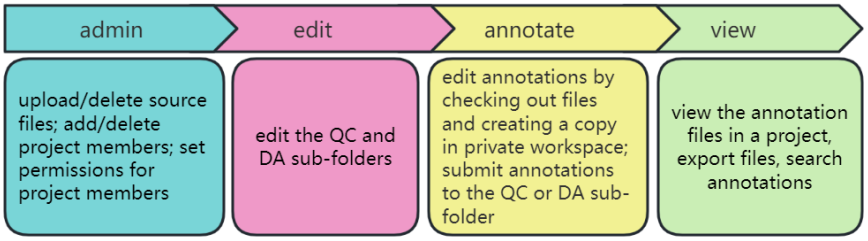

Figure 3: Permission hierarchy

and the descriptions for each are shown in Figure 3. The permission level decreases sequentially from left to right. The permission on the left side by default has all permissions on the right side. Thus, users with the "annotate" permission have the "view" permission. Similarly, users with the "edit" permission also have both the "view" and the "annotate" permissions, and so on. Same-level permission can be issued to multiple users except for "admin", which belongs to the owner of the project only.

Moreover, we have established an efficient workflow for each project:

- Each user can create project folders with the "admin" permission. "Admin" adds members to the project, assigns permissions, and uploads files into project folders. The annotation files can be exported files including UMR graphs, or just raw text.
- Anyone can view the original annotation files. Members with the "annotate" permission or higher can check out files and independently edit annotations without impacting other members in the project who have checked out the same files.
- If multiple users check out the same file, they should submit their annotations into the "DA" folder[3] after completing their work. Members with the "edit" or "admin" permission can decide which annotation should be put into the final "QC" folder by deleting the rest. If a file is checked out by only one member, the member can directly upload it to the "QC" folder. Members with the "edit" or "admin" permission can delete files with poor quality.

This workflow has been successfully applied to many projects such as the THYME corpus (Albright et al., 2013), and the Arabic UMR corpus.

Furthermore, UMR-Writer 2.0 provides a search functionality that allows users to search for annotations based on strings, concepts, or triples. Users can also specify whose annotations they want to query by entering user names. Each user has the option to choose the visibility of their project using a slider bar. The annotations in the project are publicly searchable by any user if the slider bar is checked. Annotation managers can leverage the search functionality to check annotations for beginner annotators during the training process.

5 Conclusion

UMR-Writer is a significant annotation tool for Uniform Meaning Representation. This paper introduces a new keyboard interface that constructs UMR graphs by entering index-based commands. This increases efficiency and guarantees higher accuracy of annotations. The new interface also solves the existing issues within the original click-based interface for the tool such as non-sequential tokens as concepts and variable inconsistencies across languages. The keyboard interface is especially welcomed by annotators who annotated with the AMR editor. Moreover, we introduce an annotation project workflow that can manage annotation projects efficiently.

Limitations

Raw text can sometimes be incorrectly segmented, especially in languages like Chinese which often have propagated segmentation errors due to the absence of explicit word boundaries and ambiguity caused by multi-character words with shared components. Annotators currently correct segmentation errors by using an underscore "_" to concatenate or splice tokens. However, this can be inconvenient, as annotators need to use this approach every time they edit concatenated/sliced concepts or add an edge between other concepts. In the future, we plan to allow annotators to manually correct segmentation errors by deleting or adding a space in the raw

[3]One file can be checked out by multiple users rather than just "double" annotated.

text.

Although the click-based interface supports low-resource languages well, we have not extensively experimented on many low-resource languages using the keyboard interface. Currently, for morphologically-complex languages, annotators need to manually count the index of a character in the token. Below is an example of Arapaho[4]:

(2) ceesisnoo'oebiicitiit
ceesis-noo'oe-**biicitii**-t

IC.begin-around-bead.s.t.-3S

'She is starting to bead around it.'

Here "*biicitii* " ("bead s.t.") is a concept. To select the token representing the concept, we need to input "`x1_13:20`" to represent "*biicitii*", where "`x1`" represents the token "*ceesisnoo'oebiicitiit*" as the entire sentence is a single token, "`_13:20`" represents the substring "*biicitii*" spanning from the thirteenth to the twentieth character in the token. Many low-resource languages such as Arapaho lack the lexical frame, thus we define frameworks for both non-lexicalized and lexicalized annotation of predicates and semantic roles (Gysel et al., 2021). For non-lexicalized UMR predicates, the role annotation is based on a general inventory of core participant roles given in Table A1. We are expanding the lexical frame coverage and constructing the predicate-specific definitions on the fly. The lexical entries should be mapped to the non-lexicalized roles in Table A1. We are also working on simplifying the process of selecting tokens by combining span selection with a mouse.

In the keyboard interface, the index-based variable system assigns different variables to within-sentence co-reference entities due to their distinct token alignments. Previously, we marked within-sentence co-reference using re-entrancy with the same variable. In the keyboard interface, it is necessary to identify and mark the two variables representing the co-referenced entities.

The identification of event-related concepts is crucial for annotating participant roles, as well as aspect and modality annotations. Currently, we do not include such a feature to detect eventive concepts. We plan to develop a system capable of detecting eventive concepts and providing auto-complete reminders to assist annotators in fully annotating the UMR graph. This approach aims to prevent any necessary annotations (such as the aspect of the eventive concept) from being omitted during the annotation process.

We also plan to refactor our code into a JavaScript framework, such as React.js, in a future version release. Additionally, we plan to make some improvements and changes to streamline the user experience, such as adjusting the visualization of the document-level annotation and implementing auto-completion of commands. Finally, we are currently working on mapping the named entities hierarchy in UMR to the ontology hierarchy in Wikidata.

[4]IC = initial change, 3S = third person singular.

Ethics Statement

We did not identify any potential ethical issues with the annotation tool.

Acknowledgements

This work is supported by grants from the CNS Division of National Science Foundation (Awards no: NSF_2213805, NSF_2213804, NSF_IIS 1764048, NSF_1763926 RI) entitled "Building a Broad Infrastructure for Uniform Meaning Representation" and "Developing a Uniform Meaning Representation for Natural Language Processing", respectively. We also gratefully acknowledge the support of NIH: 5R01LM010090-09, THYME-3, Temporal Relation Discovery for Clinical Text, (sub from Harvard Children's). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NSF, NIH, or the U.S. government. We extend our thanks to Jie Cao and Kathryn Conger for their valuable suggestions on the paper revision.

References

Daniel Albright, Arrick Lanfranchi, Anwen Fredriksen, IV Styler, William F, Colin Warner, Jena D Hwang, Jinho D Choi, Dmitriy Dligach, Rodney D Nielsen, James Martin, Wayne Ward, Martha Palmer, and Guergana K Savova. 2013. Towards comprehensive syntactic and semantic annotations of the clinical narrative. *Journal of the American Medical Informatics Association*, 20(5):922–930.

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*, pages 178–186, Sofia, Bulgaria. Association for Computational Linguistics.

Julia Bonn, Skatje Myers, Jens E. L. Van Gysel, Lukas Denk, Meagan Vigus, Jin Zhao, Andrew Cowell, William Croft, Jan Hajič, James H. Martin, Alexis Palmer, Martha Palmer, James Pustejovsky, Zdenka Urešová, Rosa Vallejos, and Nianwen Xue. 2023. Mapping AMR to UMR: Resources for adapting existing corpora for cross-lingual compatibility. In *Proceedings of the 21st International Workshop on Treebanks and Linguistic Theories (TLT, GURT/SyntaxFest 2023)*, pages 74–95, Washington, D.C. Association for Computational Linguistics.

Yuen Ren Chao. 1968. *A grammar of spoken Chinese / by Yuen Ren Chao*. University of California Press Berkeley.

Wei-Te Chen and Will Styler. 2013. Anafora: A web-based general purpose annotation tool. In *Proceedings of the 2013 NAACL HLT Demonstration Session*, pages 14–19, Atlanta, Georgia. Association for Computational Linguistics.

Lucia Donatelli, Michael Regan, William Croft, and Nathan Schneider. 2018. Annotation of tense and aspect semantics for sentential AMR. In *Proceedings of the Joint Workshop on Linguistic Annotation, Multiword Expressions and Constructions (LAW-MWE-CxG-2018)*, pages 96–108, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Michael Wayne Goodman. 2020. Penman: An open-source library and tool for AMR graphs. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 312–319, Online. Association for Computational Linguistics.

Jens Van Gysel, Meagan Vigus, Jayeol Chun, Kenneth Lai, Sarah Moeller, Jiarui Yao, Timothy J. O'Gorman, Andrew Cowell, W. Bruce Croft, Chu-Ren Huang, Jan Hajic, James H. Martin, Stephan Oepen, Martha Palmer, James Pustejovsky, Rosa Vallejos, and Nianwen Xue. 2021. Designing a uniform meaning representation for natural language processing. *KI - Künstliche Intelligenz*, 35:343 – 360.

Ulf Hermjakob. 2013. Amr editor: A tool to build abstract meaning representations. *USC Information Sciences Institute, Tech. Rep*.

Robert T. Kasper. 1989. A flexible interface for linking applications to Penman's sentence generator. In *Speech and Natural Language: Proceedings of a Workshop Held at Philadelphia, Pennsylvania, February 21-23, 1989*.

Bin Li, Yuan Wen, Weiguang Qu, Lijun Bu, and Nianwen Xue. 2016. Annotating the little prince with Chinese AMRs. In *Proceedings of the 10th Linguistic Annotation Workshop held in conjunction with ACL 2016 (LAW-X 2016)*, pages 7–15, Berlin, Germany. Association for Computational Linguistics.

Tim O'Gorman, Michael Regan, Kira Griffitt, Ulf Hermjakob, Kevin Knight, and Martha Palmer. 2018. AMR beyond the sentence: the multi-sentence AMR corpus. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3693–3702, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

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

Sameer Pradhan, Julia Bonn, Skatje Myers, Kathryn Conger, Tim O'gorman, James Gung, Kristin Wrightbettner, and Martha Palmer. 2022. PropBank comes of Age—Larger, smarter, and more diverse. In *Proceedings of the 11th Joint Conference on Lexical and Computational Semantics*, pages 278–288, Seattle, Washington. Association for Computational Linguistics.

James Pustejovsky, Ken Lai, and Nianwen Xue. 2019. Modeling quantification and scope in Abstract Meaning Representations. In *Proceedings of the First International Workshop on Designing Meaning Representations*, pages 28–33, Florence, Italy. Association for Computational Linguistics.

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. 2012. brat: a web-based tool for NLP-assisted text annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 102–107, Avignon, France. Association for Computational Linguistics.

Jens E. L. Van Gysel, Meagan Vigus, Lukas Denk, Andrew Cowell, Rosa Vallejos, Tim O'Gorman, and William Croft. 2021. Theoretical and practical issues in the semantic annotation of four indigenous languages. In *Proceedings of the Joint 15th Linguistic Annotation Workshop (LAW) and 3rd Designing Meaning Representations (DMR) Workshop*, pages 12–22, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jens E. L. Van Gysel, Meagan Vigus, Pavlina Kalm, Sook-kyung Lee, Michael Regan, and William Croft. 2019. Cross-linguistic semantic annotation: Reconciling the language-specific and the universal. In *Proceedings of the First International Workshop on Designing Meaning Representations*, pages 1–14, Florence, Italy. Association for Computational Linguistics.

Meagan Vigus, Jens E. L. Van Gysel, and William Croft. 2019. A dependency structure annotation for modality. In *Proceedings of the First International Workshop on Designing Meaning Representations*, pages 182–198, Florence, Italy. Association for Computational Linguistics.

Meagan Vigus, Jens E. L. Van Gysel, Tim O'Gorman, Andrew Cowell, Rosa Vallejos, and William Croft. 2020. Cross-lingual annotation: a road map for low-

and no-resource languages. In *Proceedings of the Second International Workshop on Designing Meaning Representations*, pages 30–40, Barcelona Spain (online). Association for Computational Linguistics.

Jiarui Yao, Haoling Qiu, Bonan Min, and Nianwen Xue. 2020. Annotating Temporal Dependency Graphs via Crowdsourcing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5368–5380, Online. Association for Computational Linguistics.

Jin Zhao, Nianwen Xue, Jens Van Gysel, and Jinho D. Choi. 2021. UMR-writer: A web application for annotating uniform meaning representations. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 160–167, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

A Appendix

Figure A1 is a UMR graph example borrowed from Zhao et al. (2021). It includes three sentences:

1. "Edmund Pope tasted freedom today for the first time in more than eight months."
2. "Pope was convicted on spying charges and sentenced to 20 years in a Russian prison."
3. "He denied any wrongdoing."

Each sentence is represented as a Directed Acyclic Graph (DAG), and multiple sentences can be connected to form a more complex graph at the document level.

Table A1 presents a general inventory of non-lexical core participant roles for low-resources languages.

Figure A2 is an example of project management (the "DA" folder is not shown here).

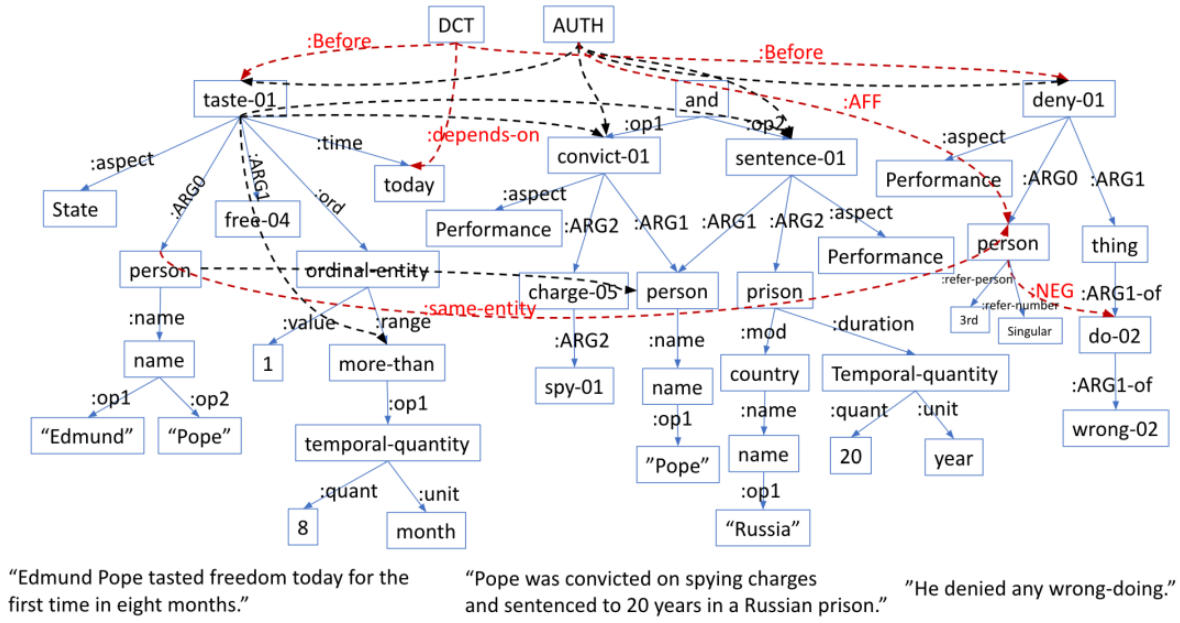

Figure A1: An example of UMR

Central roles	Actor, Undergoer, Theme, Recipient, Force, Causer, Experiencer, Stimulus
Peripheral roles	Instrument, Companion, Material/Source, Place, Start, Goal, Affectee
Roles for entities and events	Cause, Manner, Reason, Purpose, Temporal, Extent

Table A1: UMR non-lexical roles

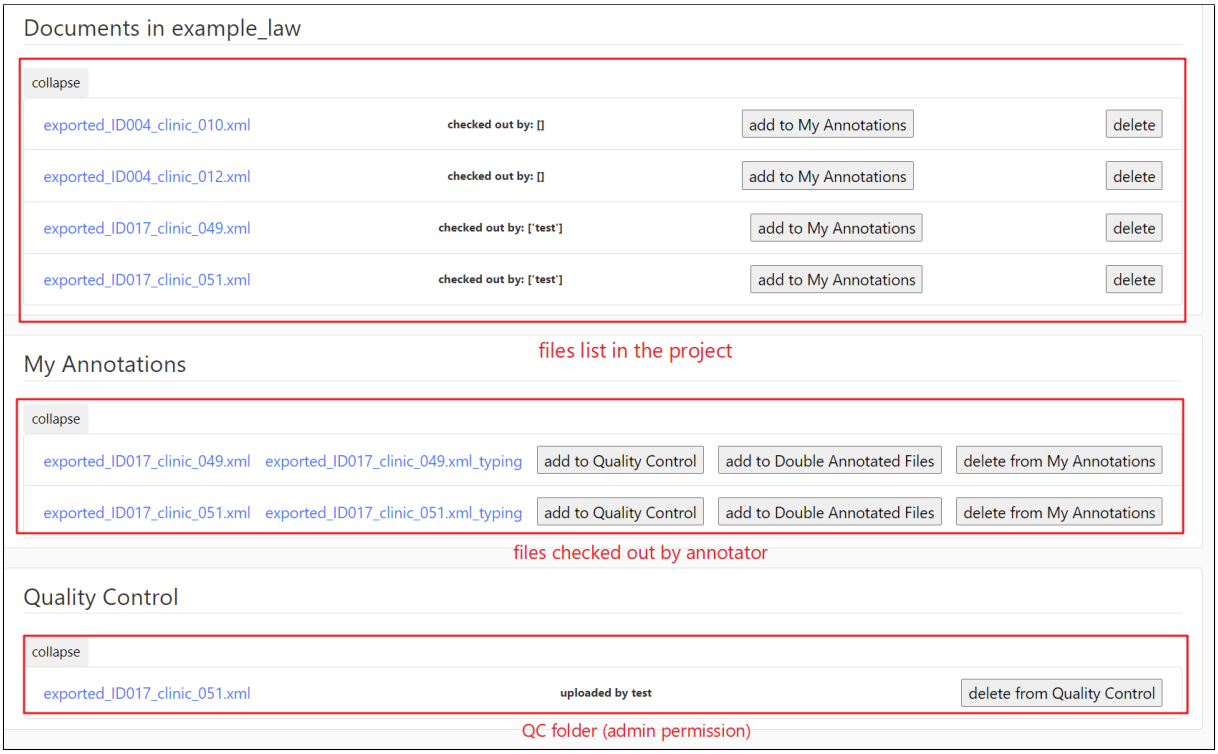

Figure A2: Project management page