UMR-Writer: A Web Application for Annotating Uniform Meaning Representations

Jin Zhao[1], **Nianwen Xue**[1], **Jens Van Gysel**[2], **Jinho D. Choi**[3]
[1]Brandeis University [2]The University of New Mexico [3]Emory University
{jinzhao,xuen}@brandeis.edu
jelvangysel@unm.edu
jinho.choi@emory.edu

Abstract

We present UMR-Writer, a web-based application for annotating Uniform Meaning Representations (UMR), a graph-based, cross-linguistically applicable semantic representation developed recently to support the development of interpretable natural language applications that require deep semantic analysis of texts. We present the functionalities of UMR-Writer and discuss the challenges in developing such a tool and how they are addressed.

1 Introduction

1.1 UMR Overview

Uniform Meaning Representation is a graph-based cross-linguistically applicable semantic representation that was recently developed with the goal of supporting interpretable natural language applications that require deep semantic analysis of texts (Van Gysel et al., 2021). UMR has two components: a sentence-level representation that is adapted from Abstract Meaning Representation (AMR) (Banarescu et al., 2013), and a document-level representation that captures semantic relations that potentially go beyond sentence boundaries. Like AMR, the UMR sentence-level representation captures the argument structures of predicative events, word senses, as well as semantic types of named entities. It also adds a representation for aspect and quantifier scope, which are not part of AMR. At the document level, UMR represents temporal (Zhang and Xue, 2018b,a; Yao et al., 2020) and modal dependencies (Vigus et al., 2019) as well as coreference. UMR abstracts away from syntactic representations and preserves semantic relations within and across sentences. Building a corpus of UMRs could potentially be very useful to NLP practitioners in multiple fields, such as information extraction and machine translation.

Figure 1 is an example UMR for a short text snippet. Like AMR, UMR is a node- and edge-labeled directed graph, where the nodes represent semantic *concepts* (including word senses, entity types etc.) and edges represent *relations* (participant roles and general semantic relations). The solid lines represent sentence-level relations while the dashed lines represent semantic relations that go beyond sentence boundaries. The direction of the arrows is always from parent to child, at both the sentence- and document level. For instance, at the sentence level, *taste-01* is an eventive concept labeled with the first sense of the lemma "taste" as defined in PropBank (Palmer et al., 2005), and a *person* concept with the name "Edmund Pope" is its *ARG0*. The concept *taste-01* also has an aspect attribute with the value *State*. The pronoun "he" in the third sentence is decomposed into a *person* concept with a person attribute *3rd* and number attribute *Singular*, indicating third person singular. At the document level, the *person* concept mapped from the pronoun "he" refers to the same entity as the *person* concept in the first sentence, as indicated by the dashed line connecting these nodes. The event *taste-01* in the first sentence occurs before document creation time (DCT), as indicated by the dashed line with the red *:Before* label, and in the third sentence, the edge label *:NEG* indicates that Edmund Pope (who corresponds to the *person* node) as a conceiver/source has a full negative epistemic stance (Boye, 2012) towards the *do* event.

1.2 Challenges in Building a Tool for UMR Annotation

As should be clear from the UMR example in Figure 1, UMR is a fairly complex representation that has many dimensions, and we need to address a number of challenges in order to develop a tool that makes UMR annotation practical. First of all, the UMR annotation scheme involves both closed and open vocabularies. For instance, while the relations, attributes, and abstract concepts (e.g., entity types such as *person*) can be selected from a closed set with a few hundred items, sense-disambiguated

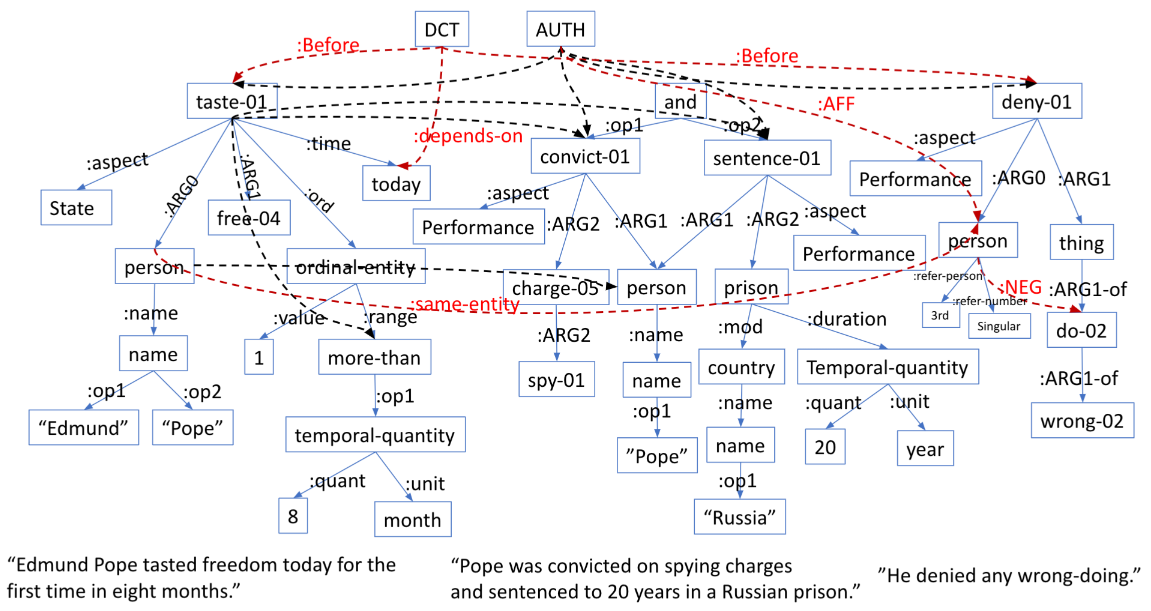


Figure 1: Uniform Meaning Representation

words (e.g., *taste-01*, *convict-01*) form open classes that need to be stored in a lexicon that can be dynamically updated during the annotation. This means that UMR-Writer needs to store both types of annotation resources to support UMR annotation and arrange them in a way that is convenient for annotators to access. Second, as UMR is a graph and thus a highly structured annotation object, UMR-Writer needs to enforce the well-formedness of UMR during the annotation process and does not leave this responsibility to the users. Similarly, UMR-Writer also needs to keep track of the variables that are crucial to the coreference aspect of the UMR annotation and automatically generates and updates the variables in response to user input.

Annotating document-level UMR adds to the complexity of an UMR annotation tool. Like any document-level annotation tool, UMR-Writer needs to present the entire document, which can be arbitrarily long. To make UMR annotation tractable and promote annotation consistency, UMR-Writer imposes an annotation procedure in which the user will proceed in a sentence-by-sentence manner. Another challenge in UMR annotation is that the UMR document-level representation, i.e., temporal and modal dependencies and coreference, needs to make reference to variables for sentence-level concepts. There is also the need sometimes for the user to make corrections to the sentence-level annotation that will affect the well-formedness of the document-level annotation. We design UMR-Writer in a way that any changes made at the sentence-level will result in an automatic update of the document-level UMR if necessary.

As the name suggests, UMR is intended to be a semantic representation uniform across languages, and UMR-Writer needs to support multi-lingual annotation and address the challenges resulting from this need. At a very basic level, UMR-Writer needs to support the display of various writing systems for languages of the world, and this factored heavily into our decision to develop a web-based tool that handles multilingual functionalities by piggy-backing on the web-browsers. Languages are also diverse with regard to their linguistic features, and the amount of linguistic resources available. For instance, some languages are morphologically complex while other languages are morphologically simpler. In terms of data sources, some languages have data from formal genres with well-defined sentence boundaries while other languages only have transcriptions of oral recordings where sentence boundaries are not always as clear. In terms of availability of annotation support resources, high-resource languages like English and Chinese have well-developed lexical resources like PropBank frame files through years of research (Palmer et al., 2005; Xue and Palmer, 2009; Xue, 2006) while low-resource languages may not have frame files at all. UMR-Writer needs to be flexible and allow this variability. To cope with languages that have no linguistic resources at their disposal, UMR-Writer allows UMR annotation of languages without pre-existing computational resources such as frame files or digital lexicons by providing a lexicon-building feature that aids in the development of linguistic resources as UMRs are annotated. To assist less experienced annotators, UMR-Writer presents PropBank frames with argument structure information for each lemma (when available), inferred from the surface forms of a word the user selects. To promote annotation consistency, UMR-Writer only allows the user to select UMR relations from a pre-specified list. This eliminates invalid concepts and roles in annotated UMRs.

2 Related Work

Fundamentally, UMR is a representation based on relations between concepts, and there are a number of tools that support annotation for relations. Some examples include Anafora (Chen and Styler, 2013), MAE (Stubbs, 2011; Rim, 2016), WebAnno (Eckart de Castilho et al., 2016), and BRAT (Stenetorp et al., 2012). Anafora is a web-based tool that supports the annotation of relations between text spans. MAE is a standalone tool that offers flexible and versatile schema support for complex relation sets. WebAnno supports semantic role labelling or event annotations, and it enables the annotation of semantic structures and the handling of rich semantic tagsets. BRAT provides intuitive annotation visualization to help users understand the relations between text annotations. However, all of these tools only support annotation based on text spans, and not annotation that requires transforming word tokens in the source text into concepts in the annotated graph that take the form of (sense-disambiguated) word lemmas, concatenated words, or even abstract concepts that do not correspond to any specific word token in the source text. In contrast, UMR-Writer allows the creation of concepts that are transformations from word tokens in the source text or purely new additions.

AMR Editor (Hermjakob, 2013) is a tool created for AMR annotation and is most similar to UMR-Writer. Like UMR-Writer, it also supports the annotation of concepts that are different from word tokens in the source text. It also makes use of both closed sets of abstract concepts and relations as well as open-class lexicons. However, it offers limited support for languages other than English, and does not support document-level annotation.

3 System Overview

UMR-Writer is implemented in JavaScript interacting with HTML pages, and uses Flask as the server side web framework. It is deployed at Heroku with a Postgres database at the backend to store annotated UMR graphs.[1]

UMR-Writer provides a Graphical User Interface (GUI) that allows annotators to point and select words from the source text, and then select and click to add concepts and relations to the UMR graph. UMR-Writer has separate views for sentence-level and document-level annotation, but the two views share the same underlying data structure. At the sentence level, UMR-Writer makes clear distinctions between the annotation of lexicalized and abstract concepts, named entity types, attributes, and relations. At the document level, UMR-Writer has separate functionalities for annotating temporal and modal dependencies as well as coreference. UMR-Writer allows the user to easily switch between the sentence-level and document-level views with a simple click.

3.1 Importing Source text into UMR-Writer for Annotation

Annotators can upload their source data in the form of single files for annotation from the upload page. UMR-Writer can parse and render plain text format as well as 6 output format variations of FLEx and Toolbox, tools commonly used by field linguists. In addition, the user can also upload an output file exported by UMR-Writer to add more annotation or make corrections. UMR-Writer extracts and retains all information in the imported source file useful for UMR annotation (e.g., morphological segmentation, word-level and morpheme-level glosses, paragraph boundaries, etc.). This is particularly important for field linguists annotating languages that they are not native speakers of. Users have access to all files they have uploaded in their individual accounts. A short sample text comes with every newly registered account for new users to try out the the tool without having to first upload their own data.

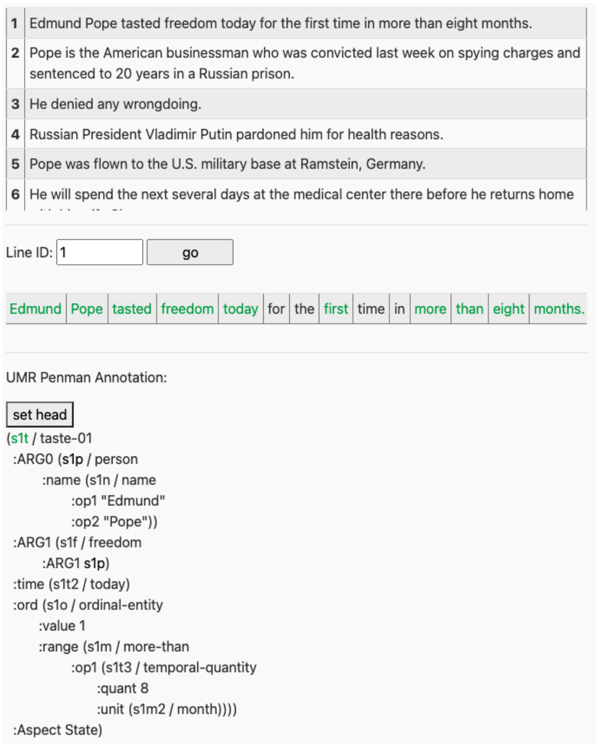
The upload page also has functionality that allows PropBank-style frame files (Palmer et al.,

[1]https://github.com/jinzhao3611/umr-annotation-tool

2005; Xue and Palmer, 2009) to be imported into UMR-Writer to support the annotation of word senses and lexicalized semantic roles. So far, English and Chinese frame files have already been pre-loaded into UMR-Writer. Annotators for other languages can upload their own lexicon as support data so long as it is in the FLEx export format.

3.2 The View for Sentence-level UMR Annotation

In this view, the user can iteratively build up the sentence-level UMR graph and UMR-Writer will automatically record the alignment between the UMR concepts and word tokens in the source text. The UMR graph is rendered in PENMAN notation (Kasper, 1989; Goodman, 2020).



1	Edmund Pope tasted freedom today for the first time in more than eight months.
2	Pope is the American businessman who was convicted last week on spying charges and sentenced to 20 years in a Russian prison.
3	He denied any wrongdoing.
4	Russian President Vladimir Putin pardoned him for health reasons.
5	Pope was flown to the U.S. military base at Ramstein, Germany.
6	He will spend the next several days at the medical center there before he returns home

Line ID: 1 go

Edmund	Pope	tasted	freedom	today	for	the	first	time	in	more	than	eight	months.

UMR Penman Annotation:

set head

```
(s1t / taste-01
 :ARG0 (s1p / person
       :name (s1n / name
             :op1 "Edmund"
             :op2 "Pope"))
 :ARG1 (s1f / freedom
       :ARG1 s1p)
 :time (s1t2 / today)
 :ord (s1o / ordinal-entity
      :value 1
      :range (s1m / more-than
             :op1 (s1t3 / temporal-quantity
                  :quant 8
                  :unit (s1m2 / month))))
 :Aspect State)
```

Figure 2: Sentence-level annotation

3.2.1 Building the Sentence-level UMR Graph

The sentence-level UMR graph building process starts with the user choosing a concept as the root node of the graph. After that, the user iteratively builds up the UMR graph by setting the parent, choosing either a lexicalized or abstract concept as its child, selecting a relation between the child and the parent, and adding this "triple" to the UMR graph. Alternatively, the user can also select an attribute for the parent, set its value, and add it to the UMR graph. Annotators can set the parent by double-clicking the node in the partially completed graph. In case corrections need to be made to the UMR graph, the user can enter the editing/deleting mode and modify the graph directly. Possible corrections include making changes to concepts or deleting subgraphs from the UMR graph.

Annotating lexicalized or abstract concepts As seen in Figure 2, the view of the sentence-level annotation varies based on the imported data. Minimally, UMR-Writer presents a single tokenized sentence for the user to annotate. When available, morphemes and their glosses can also be displayed to support UMR annotation. For example, in morphologically complex languages like Sanapaná (Enlhet-Enenlhet, Paraguay), information about which morpheme within a word is the root is crucially important to help annotators choose a lemma form as the UMR concept. When a text span in the sentence is selected, UMR-Writer automatically generates a few concept options for the user to choose from. These options include word senses (if the selected text matches a word in the frame files), a lemma form of the word (when there is no matching entry in the frame files), or a concatenated form (when multiple words are selected in the text span). Annotators can also choose an *abstract* concept from a pre-defined drop-down menu that does not map to any word token but can be inferred from the context. Finally, UMR-Writer provides short-hand buttons for adding named entity concepts to the UMR graph. Named entity concepts have internal structures that are predictable, but usually take several actions to complete.

Annotating semantic relations The annotator can select a semantic relation that relates a concept to its parent in the UMR graph using a "Roles" menu. The range of semantic relations includes lexicalized participant roles such as *:ARG0*, non-lexicalized roles such as *:agent*, as well as other semantic relations that are not typically considered participant roles (e.g., *:poss*, *:name*). More information on where each set of participant roles is used can be found in the UMR guidelines[2].

Annotating UMR attributes Annotators can choose the attribute type (e.g. *:Aspect*) from an "Attributes" menu, and choose the corresponding attribute value in a pop-up "Attribute Values" menu

[2]https://umr4nlp.github.io/web

to add the attribute to the UMR graph. For instance, the values of *:Aspect* include both more fine-grained values such as *State* and more coarse-grained values such as *Atelic Process*, which are organized in a lattice (Van Gysel et al., 2021). This makes the tool more cross-linguistically general, because some languages lack overt aspectual marking in their grammar, making fine-grained values hard to distinguish, while in other languages more fine-grained distinctions are overtly marked.

3.2.2 Token-Concept Alignments

As the user selects a text span to create a lexicalized concept, UMR-Writer automatically records an alignment between the text span and the concept in the UMR graph. As it is possible for some concepts to be created out of more than one word token or part of a word token in the source sentence, or even no lexical material at all in the case of abstract concepts, the mapping between word tokens and UMR concepts will not be one-to-one. This alignment is potentially useful for purposes of improving UMR parsing accuracy or for linguists who wish to study syntax-semantic mismatches.

3.3 View for Document-level Annotation

For both sentence-level and document-level annotation, we assume an annotation procedure in which a document is annotated sentence by sentence. The sentence-level representation is annotated first, so that the document-level annotation can make reference to the concepts and relations in the sentence-level representation (Van Gysel et al., 2021). An integrated document-level view of UMR-Writer is shown in Figure 3: the completed sentence-level annotations are displayed on the left, and the document-level annotations in the middle column are created by linking all child concepts in the current sentence to a parent in the previous or in some cases the following sentences. To do temporal annotation (Van Gysel et al., 2021), the user selects a child and a parent which can be either an eventive or time concept, and then identifies the relation (e.g., *:Before*, *:After*) between them based on contextual clues. Similarly, when annotating modal relations, the annotator can select a parent and a child and identify a relation that captures the epistemic strength and polarity (e.g., *:AFF*, *:NEG*) between the parent and child. For coreference annotation, the annotator determines if the parent and the child refer to the same entity/event or one designates a subset of the other.

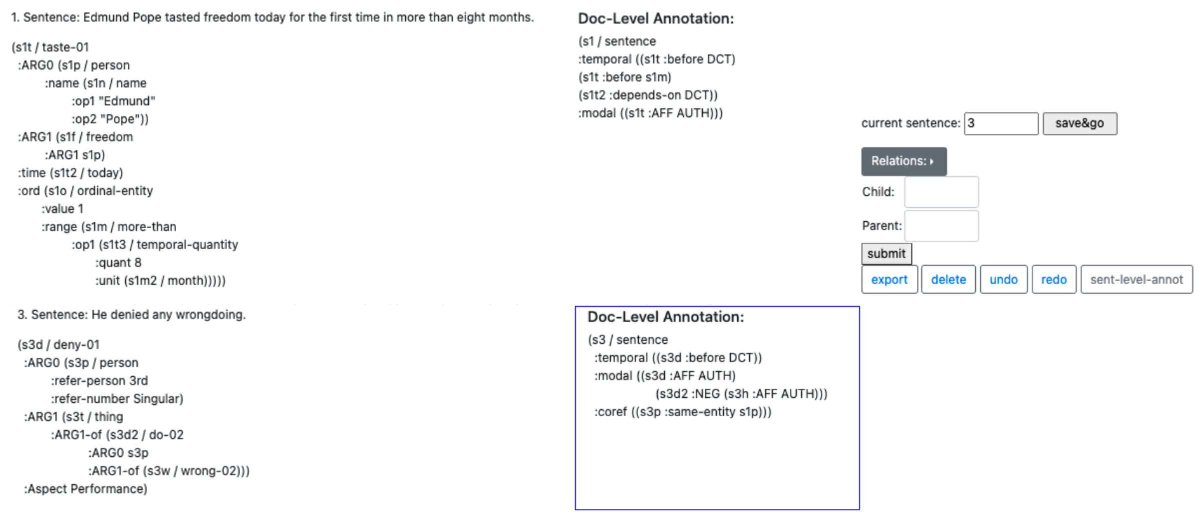
The document-level and sentence-level view of UMR-Writer are tightly integrated in the sense that any change in the sentence-level graph results in the automatic update of the document-level graph. This way, the burden of ensuring the integrity of the UMR graph is shifted away from the user. This is achieved by storing all the sentence-level UMR graphs for a document in a single data structure. When the user updates a node in a sentence-level UMR graph, all document-level annotations making reference to that node will be updated as well.

3.4 Support for Cross-lingual Annotation

A sentence-by-sentence annotation procedure at both the sentence- and document-level is appealing in that it makes the annotation more tractable for the user and potentially for models trained on the resulting annotated UMRs. However, data from some languages cannot be cleanly segmented into "sentences" that allow us to make the simplifying one-line-per-sentence assumption when implementing UMR-Writer. This is typically the case for data for low-resource languages collected by field linguists, who often segment and transcribe the recordings from their fieldwork by intonation units. The example in (1) shows the English gloss of three Sanapaná intonation units from an oral history recording (Van Gysel et al., 2020). Semantically, they form one predicate-argument complex, but as they were not uttered under a single intonation contour, they would be represented as three lines in the text.

(1) a. Then the cuartelero bird went to eat.
 b. In the lagoon.
 c. Fish, and eels.

If we strictly follow the one-line-per-sentence assumption during sentence-level annotation, in order to capture the semantic relation between the "eat" concept in (1a) and the "fish" concept in (1c), we would have to posit an abstract, implicit concept as its *patient* when annotating "eat", and later link it to the "fish" concept during document-level annotation via coreference between these concepts. We also need to do the same for the relation between "eat" and "lagoon". This would be easy to implement but cumbersome to the user. On the other hand, if we allow the user to annotate semantic relations from lines that are arbitrarily distant, this would make implementation intractable and error-prone. We adopt a compromise and allow the user to annotate semantic relations between two adjacent lines when annotating sentence-level UMR.


1. Sentence: Edmund Pope tasted freedom today for the first time in more than eight months.

```
(s1t / taste-01
 :ARG0 (s1p / person
       :name (s1n / name
             :op1 "Edmund"
             :op2 "Pope"))
 :ARG1 (s1f / freedom
       :ARG1 s1p)
 :time (s1t2 / today)
 :ord (s1o / ordinal-entity
      :value 1
      :range (s1m / more-than
             :op1 (s1t3 / temporal-quantity
                  :quant 8
                  :unit (s1m2 / month)))))
```

Doc-Level Annotation:

```
(s1 / sentence
:temporal ((s1t :before DCT)
(s1t :before s1m)
(s1t2 :depends-on DCT))
:modal ((s1t :AFF AUTH)))
```

current sentence: 3

3. Sentence: He denied any wrongdoing.

```
(s3d / deny-01
  :ARG0 (s3p / person
        :refer-person 3rd
        :refer-number Singular)
  :ARG1 (s3t / thing
        :ARG1-of (s3d2 / do-02
                 :ARG0 s3p
                 :ARG1-of (s3w / wrong-02)))
  :Aspect Performance)
```

Doc-Level Annotation:

```
(s3 / sentence
  :temporal ((s3d :before DCT))
  :modal ((s3d :AFF AUTH)
              (s3d2 :NEG (s3h :AFF AUTH)))
  :coref ((s3p :same-entity s1p)))
```

Figure 3: Document-level-annotation

This relieves the burden from the user to a large extent while making the implementation tractable.

This example also illustrates the need to present multiple sentences even for sentence-level annotation to provide enough context for the user to understand and properly annotate a sentence. Even in the sentence-level view, UMR-Writer presents all sentences in the document to serve as context for the sentence being currently annotated.

3.4.1 Dynamic Updating of the Annotation Lexicon

When a user creates UMR concepts from spans of text in the source sentence, UMR-Writer suggests possible concepts by lemmatizing the word token and using the resulting lemma to query the frame file lexicon and retrieve a list of senses for this lemma as well as the semantic roles associated with each sense. When this lemma is not in the lexicon, UMR-Writer suggests using the lemma as the concept. While lemmatization is relatively straightforward and can be done algorithmically for languages like English and Chinese, it cannot be reliably done for morphologically complex low-resource languages. As a result, UMR-Writer cannot make accurate suggestions for these languages, and the user has to edit the suggested lemma before attaching it to the UMR graph. To avoid repeated editing of the same word token, UMR-Writer allows the user to enter a word token and its associated lemma with different senses and argument structure information into a database from a separate lexicon page. This way next time the user encounters the same word or the different inflected form of the same word, UMR-Writer will be able to retrieve its lemma as a suggestion. When entering the lemma for the word token, UMR-Writer can also provide the lemmas for other variations of the same word as a reminder to help the user choose the correct lemma. Different users working on the same language can also share the same lexicon.

4 How to Access UMR-Writer

UMR-Writer now has a demo version that allows users to register an account and use it for their own annotation.[3] UMR-Writer was used in annotation efforts for Kukama, Arapaho, Sanapaná, and Navajo. UMR-Writer is intended to be a tool that annotators can use to create data sets for NLP researchers to train machine learning models, and for linguists to study the semantic structures of language. We intend to make it open-source and make it available to the broader research community. This tool will be released under creative commons under version CC BY-NC. It has been tested by about 30 annotators who are field linguists, computational linguists, or other types of users with UMR annotation tasks for various languages.

[3]The question of whether multiple users can access the same document and generate multiple UMRs of the same sentences or documents came up from our reviewers. Allowing this would be useful for adjudication. For now, each annotator uploads the same source file to their account, makes their own annotation, and then shares the exported file with the other so that IAA can be calculated. Adjudication is a non-trivial task in the UMR annotation process, the specifics of which we are currently working on.

5 Future Development

It would also be useful for UMR-Writer to be made inter-operable with other platforms. For example, FLEx, used by many field linguists for language documentation, offers a convenient way of storing texts, providing morphological glosses, and linking these to the lexicon. Efforts will be made to integrate and work more seamlessly with such common field linguist tools.

We also plan to make UMR-Writer support more languages. We will continue to improve the tool to promote annotation efficiency and consistency as ease of use as we receive feedback for users.

Acknowledgment

This work is supported by a grant from the IIS Division of National Science Foundation (Awards No. 1763926) entitled “Building a Uniform Meaning Representation for Natural Language Processing” awarded to Nianwen Xue). All views expressed in this paper are those of the authors and do not necessarily represent the view of the National Science Foundation. We thank the collaborators of the UMR project for feedback during the development of this annotation tool.

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

Kasper Boye. 2012. *Epistemic meaning: A crosslinguistic and functional-cognitive study*, volume 43 of *Empirical Approaches to Language Typology*. De Gruyter Mouton, Berlin.

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.

Richard Eckart de Castilho, Éva Mújdricza-Maydt, Seid Muhie Yimam, Silvana Hartmann, Iryna Gurevych, Anette Frank, and Chris Biemann. 2016. A web-based tool for the integrated annotation of semantic and syntactic structures. In *Proceedings of the Workshop on Language Technology Resources and Tools for Digital Humanities (LT4DH)*, pages 76–84, Osaka, Japan. The COLING 2016 Organizing Committee.

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.

Ulf Hermjakob. 2013. AMR Editor: A Tool to Build Abstract Meaning Representations.

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*.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics*, 31(1):71–106.

Kyeongmin Rim. 2016. Mae2: Portable annotation tool for general natural language use. In *Proc 12th Joint ACL-ISO Workshop on Interoperable Semantic Annotation*, pages 75–80.

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.

Amber Stubbs. 2011. Mae and mai: lightweight annotation and adjudication tools. In *Proceedings of the 5th linguistic annotation workshop*, pages 129–133.

Jens E. L. Van Gysel, Roberto Álvarez, Florentino Álvarez, Valenciano Cabrera, Cano Carlos, Cecilio Teytaro, Esteban López, Regino Teytaro, and Marty Adamsmith. 2020. Roberto Álvarez talking about the history of the Sanapaná people / Roberto Álvarez contando sobre la historia del pueblo Sanapaná. In Jens E. L. Van Gysel (Collector): A documentation of historical narratives amongst the Sanapaná (Enlhet-Enenlhet) of the Paraguayan Chaco. London: Endangered Languages Archive. Session Date: 25 July 2019. Accessed: 28 June 2021.

Jens E. L. Van Gysel, Meagan Vigus, Jayeol Chun, Kenneth Lai, Sarah Moeller, Jiarui Yao, Tim O'Gorman, Andrew Cowell, William Croft, Chu-Ren Huang, Jan Hajič, 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*.

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.

Nianwen Xue. 2006. A chinese semantic lexicon of senses and roles. *Language resources and evaluation*, 40(3):395–403.

Nianwen Xue and Martha Palmer. 2009. Adding semantic roles to the chinese treebank. *Natural Language Engineering*, 15(1):143.

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.

Yuchen Zhang and Nianwen Xue. 2018a. Neural ranking models for temporal dependency structure parsing. *arXiv preprint arXiv:1809.00370*.

Yuchen Zhang and Nianwen Xue. 2018b. Structured interpretation of temporal relations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).