AMR Editor: A Tool to Build Abstract Meaning Representations

Ulf Hermjakob
USC Information Sciences Institute
4676 Admiralty Way #938
Marina del Rey, CA 90292, USA
ulf@isi.edu

Abstract

This demo paper describes a web-based editor of *Abstract Meaning Representations* (AMRs) to build semantic representations of English. It supports annotators in building a corpus of gold AMRs, which can be used in applications such as semantic parsing and semantics-based machine translation. The AMR Editor currently has about 50 active users worldwide, including at the Linguistic Data Consortium (LDC) and SDL Language Weaver, and has been used to write more than 10,000 AMRs in 2012, for sentences from newswire, web and other genres. It is publicly available at `http://www.isi.edu/~ulf/amr/AMR-editor.html` with a guest access option for users without an account.

1 Abstract Meaning Representation

The purpose of Abstract Meaning Representation (AMR) is to provide natural language applications with a deeper representation than a syntax tree, allowing for semantic language modelling, abstracting from many idiosyncrasies of English syntax and morphology, and providing a correctable internal semantic representation for semantics-based machine translation. Here is an example:

```
(w / want-01
  :ARG0 (b / boy)
  :ARG1 (e / eat-01
           :ARG0 b
           :polarity -
           :time (n / now)))
```

AMR for *The boy doesn't want to eat now.*

The AMR in the preceding example uses concepts such as *eat-01* (PropBank sense 01 for *eat*), roles such as *:ARG0* (typically used for agents in PropBank frames), and variables such as *b*, which in this example co-indexes the agent/experiencer of *want-01* with the agent of *eat-01*.

AMR has been designed in collaboration with researchers at the USC Information Sciences Institute (ISI), the University of Colorado, LDC, SDL and others to represent the semantics of a sentence in a single representation, in a simple format, practical enough to build a large-scale semantics bank, and building on existing resources such as OntoNotes (Weischedel, 2011).

Note, however, that AMR is not an Interlingua. It is geared towards English and has other limitations such as not representing quantifier scope.

See (Knight, 2012) for a more detailed description and discussion of AMR.

2 AMR Input Methods

Our research group wrote the first few dozen AMRs in a regular text editor, but we quickly experienced numerous problems. Parentheses did not match, variables were "double-booked", changes were cumbersome, and resources such as lists of valid roles and OntoNotes were not easily accessible, so we soon saw the need for an AMR Editor.

To support annotators at different levels of experience, the AMR Editor offers several input methods and provides strong support to choose the proper concepts, roles and overall structure.

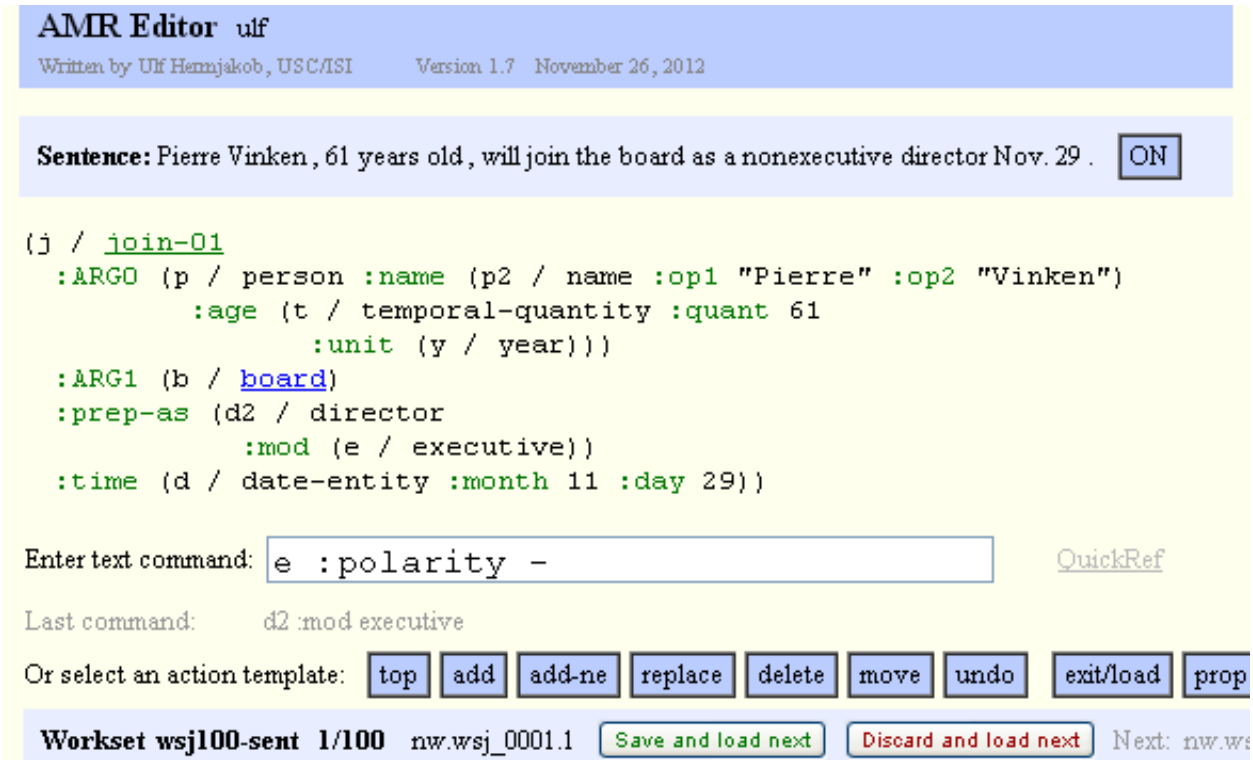

Figure 1: Screenshot of the AMR Editor when entering a text command, showing the core portion of the main window.

2.1 Templates

The editor offers a number of ways to write AMRs. All core editing operations offer templates that the annotator can fill out. For example, to start an AMR, click on the *top* action box and then enter the top concept level *want*. To grow the AMR, click on the *add* action box to enter *w*, *:arg0*, and *boy* respectively. Many boxes, fields and other objects in the AMR Editor have tooltip support for further guidance. For an example screenshot using template-based editing, see Figure 3.

2.2 Point and Click

The editor also offers point-and-click editing for delete and replace operations, where the annotator can use the mouse to select from the AMR the part to be deleted or changed. These input methods make editing easy for novice annotators.

2.3 Text Command

More experienced annotators tend to edit AMRs using the text command field, where they can enter short "Unix-style" editing commands such as *w :arg1 eat*. The editor supports the transition towards such more efficient editing by showing in the field following *Last command:* the text command form of the last command. For an example screenshot, see Figure 1.

2.4 Post Editing

We developed scripts to automatically generate AMRs from English, and for sentences covered by OntoNotes, the quality of those AMRs is currently that of a beginning-to-intermediate-level human annotator. Annotators have the option to build gold AMRs by post-editing such automatically generated AMRs.

3 Using OntoNotes

AMR builds on OntoNotes. For verb senses, it adopts PropBank senses and core frame argument structures. The AMR Editor will recognize concepts such as *eat* as covered by OntoNotes and will underline such concepts. Clicking on such underlined concept will open up a new window offering a choice of senses (with frames). Clicking on a spe-

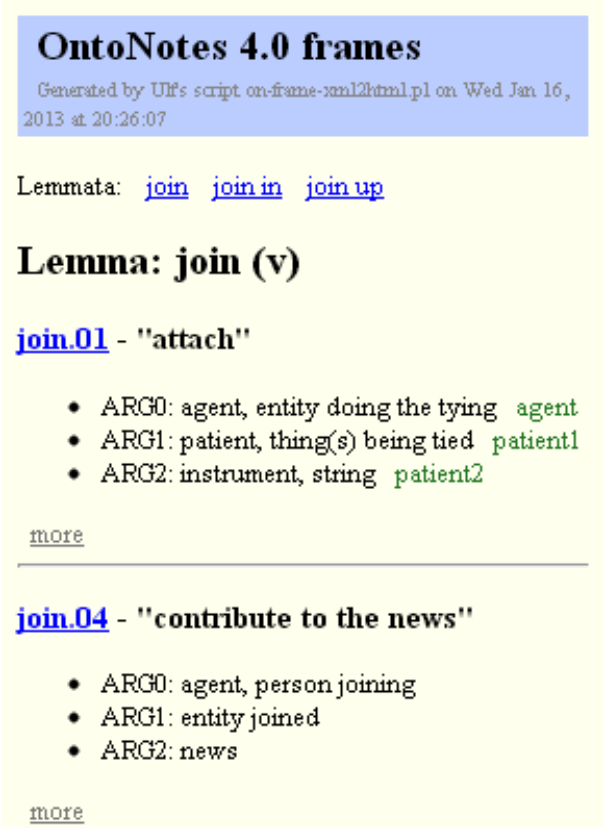

Figure 2: Screenshot of the OntoNotes/PropBank sense selection window, showing the top two choices.
Clicking on more will expand the sense selection window to show examples. Clicking on join.01 will select that sense in the main editor window (and close the selection window).

cific sense will choose that sense for the AMR in the main window. For an example screenshot, see Figure 2.

Using OntoNotes annotations, partial AMRs for named entities, dates and other quantities can be predicted with high accuracy, so the AMR Editor will offer the annotators to include such partial AMRs, thereby improving annotation speed and accuracy.

Finally, if OntoNotes annotations are available, they can be viewed as a reference in a user-friendly format by clicking on the *ON* button.

4 AMR Guidance

The AMR Editor provides a wide range of support for the annotator to choose the proper concepts, roles and overall structure.

- **AMR Editor manual**, release notes and FAQs
- a **QuickRef** link next to the *text command* box
- **AMR Guidelines** (79 pages), which can be summoned by typing "guidelines" in the *text command* box
- a list of **roles** with explanations and examples, which can be accessed in several ways:
  1. type "roles" in the *text command* box
  2. type :? at any point in the *text command*
  3. click on the *role:* box in the add/add-ne templates
- a list of **named-entity types** with explanations and examples, which can be accessed in analogy to roles
- a **search** function that returns precedents from more than 400 highly adjudicated AMRs; or alternatively AMRs from one's own or some other person's or group's AMRs, e.g. by typing in "search neither nor" into the *text command* box
- Some words and phrases of the plain English sentence are underlined, signalling underlying information that will be shown during a mouse-over (compact info) or when clicking on it (extended info with examples). For example, in

  Should I be afraid ?

  the system will suggest to consider and provide examples for using concepts like *recommend-01* (for *should*) and *fear-01* (for *afraid*), as well as a role-value pair *:mode interrogative* for questions.

5 Other Features of the AMR Editor

5.1 Workflow

For large-scale annotations, sentences are organized into *worksets* of typically about 200 sentences, with navigation buttons, a sentence menu, and a choice of three formats to export the AMRs.

5.2 Administrative Support

Annotation supervisors can create AMR Editor accounts; create and share worksets; and monitor annotation progress using an *activity report* function.

Figure 3: Screenshot of the AMR Editor when using the *add-ne* template for *Steven Spielberg lives in Los Angeles.*

5.3 Preferences

The editor's **option** box (outside of the screenshot area of Figures 1 and 3) allows the user to specify a range of display and editing preferences, for example whether or not to keep named entities on the same line (for a more compact representation).

5.4 Implementation

The AMR Editor has been implemented in JavaScript (for instant-response editing of an AMR on the HTML page on the user's computer) and Perl (for support scripts running at ISI).

5.5 Browser Settings

Since the AMR Editor is web-based, users don't need to install any software, but JavaScript and pop-up windows must be enabled for the AMR Editor.

6 Annotation Statistics

The AMR Editor has been used to annotate 5,727 corpus sentences in 2012, some by multiple annotators, resulting in over 10,000 AMRs. Sentences were largely from newswire, weblogs, fiction ("The Little Prince") and broadcast conversation. Most of the AMRs were built by 12 LDC annotators in the 4th quarter of 2012. Annotation times averaged 10 minutes per sentence, with sentences ranging from 1 to 189 words, with an average length of 19 words.

Acknowledgments

This work was supported by DARPA/IPTO Contract No. HR0011-12-C-0014 and the US Air Force (via BBN) Contract No. FA8750-09-C-0179.

References

Kevin Knight, Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Philipp Koehn, Martha Palmer, Nathan Schneider. 2012. *Abstract Meaning Representation (AMR) 1.0 Specification*. `http://www.isi.edu/~ulf/amr/help/amr-guidelines.pdf`

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, Mohammed El-Bachouti, Robert Belvin, Ann Houston. 2011. *OntoNotes Release 4.0*. Linguistic Data Consortium, Philadelphia.