DEPENDENCY PARSING, FRAMENET, SEMANTIC ROLE LABELING, SEMANTIC PARSING

Heng Ji
jih@rpi.edu
September 16, 2016

Acknowledgement: FrameNet slides from Charles Fillmore; AMR slides from Nathan Schneider, Jeff Flanigan, Tim O’Gorman
Outline

• Semantic Parsing Overview
• Dependency Parsing
  • Formal definition
  • Dynamic programming
  • Supervised Classification

• Semantic Role Labeling
  • Propbank
  • Automatic SRL
  • FrameNet

• Abstract Meaning Representation
Semantic Parsing

• “Semantic Parsing” is, Ironically, a semantically ambiguous term
  • Semantic role labeling
  • Finding generic relations in text
  • Transforming a natural language sentence into its meaning representation
Semantic Parsing

• **Semantic Parsing**: Transforming *natural language* (NL) sentences into *computer executable* complete *meaning representations* (MRs) for domain-specific applications

• Realistic semantic parsing currently entails domain dependence

• Example application domains
  • ATIS: Air Travel Information Service
  • CLang: Robocup Coach Language
  • Geoquery: A Database Query Application
ATIS: Air Travel Information Service

- Interface to an air travel database [Price, 1990]
- Widely-used benchmark for spoken language understanding

May I see all the flights from Cleveland to Dallas?

Air-Transportation
Show: (Flight-Number)
Origin: (City "Cleveland")
Destination: (City "Dallas")

Semantic Parsing

Query

NA 1439, TQ 23, …
CLang: RoboCup Coach Language

- In RoboCup Coach competition teams compete to coach simulated players [http://www.robocup.org]
- The coaching instructions are given in a computer language called CLang [Chen et al. 2003]

If the ball is in our goal area then player 1 should intercept it.

(bpos (goal-area our) (do our {1} intercept))

Semantic Parsing

Simulated soccer field
Geoquery: A Database Query Application

- Query application for U.S. geography database containing about 800 facts [Zelle & Mooney, 1996]

Which rivers run through the states bordering Texas?

answer(traverse(next_to(stateid('texas'))))

Answer: Arkansas, Canadian, Cimarron, Gila, Mississippi, Rio Grande ...

Semantic Parsing
What is the meaning of “meaning”?

• Representing the meaning of natural language is ultimately a difficult philosophical question
• Many attempts have been made to define generic formal semantics of natural language
  • Can they really be complete?
  • What can they do for us computationally?
  • Not so useful if the meaning of Life is defined as Life’
• Our meaning representation for semantic parsing does something useful for an application
• Procedural Semantics: The meaning of a sentence is a formal representation of a procedure that performs some action that is an appropriate response
  • Answering questions
  • Following commands
Meaning Representation Languages

- *Meaning representation language* (MRL) for an application is assumed to be present

- MRL is designed by the creators of the application to suit the application’s needs independent of natural language

- CLang was designed by RoboCup community to send formal coaching instructions to simulated players

- Geoquery’s MRL was based on the Prolog database
Engineering Motivation for Semantic Parsing

• Applications of domain-dependent semantic parsing
  • Natural language interfaces to computing systems
  • Communication with robots in natural language
  • Personalized software assistants
  • Question-answering systems

• Machine learning makes developing semantic parsers for specific applications more tractable

• Training corpora can be easily developed by tagging natural-language glosses with formal statements
Cognitive Science Motivation for Semantic Parsing

• Most natural-language learning methods require supervised training data that is not available to a child
  • No POS-tagged or treebank data

• Assuming a child can infer the likely meaning of an utterance from context, NL-MR pairs are more cognitively plausible training data
Distinctions from Other NLP Tasks: Deeper Semantic Analysis

- *Information extraction* involves shallow semantic analysis

Show the long email Alice sent me yesterday

<table>
<thead>
<tr>
<th>Sender</th>
<th>Sent-to</th>
<th>Type</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>Me</td>
<td>Long</td>
<td>7/10/2010</td>
</tr>
</tbody>
</table>
Distinctions from Other NLP Tasks: Deeper Semantic Analysis

- *Semantic role labeling* also involves shallow semantic analysis

```
Show the long email Alice sent me yesterday
```

Diagram:
- **Sender**
- **Recipient**
- **Theme**

- **Show** (verb)
- **the** (determiner)
- **long** (adjective)
- **email** (noun)
- **Alice** (proper noun)
- **sent** (verb)
- **me** (pronoun)
- **yesterday** (adverb)
Distinctions from Other NLP Tasks: Deeper Semantic Analysis

- Semantic parsing involves deeper semantic analysis to understand the whole sentence for some application

Show the long email Alice sent me yesterday
Distinctions from Other NLP Tasks: Final Representation

- Part-of-speech tagging, syntactic parsing, SRL etc. generate some intermediate linguistic representation, typically for latter processing; in contrast, semantic parsing generates a final representation.
Distinctions from Other NLP Tasks: Computer Readable Output

- The output of some NLP tasks, like question-answering, summarization and machine translation, are in NL and meant for humans to read.
- Since humans are intelligent, there is some room for incomplete, ungrammatical or incorrect output in these tasks; credit is given for partially correct output.
- In contrast, the output of semantic parsing is in formal language and is meant for computers to read; it is critical to get the exact output, strict evaluation with no partial credit.
Distinctions from Other NLP Tasks

• Shallow semantic processing
  • Information extraction
  • Semantic role labeling

• Intermediate linguistic representations
  • Part-of-speech tagging
  • Syntactic parsing
  • Semantic role labeling

• Output meant for humans
  • Question answering
  • Summarization
  • Machine translation
Relations to Other NLP Tasks: Syntactic Parsing

- Semantic parsing inherently includes syntactic parsing but as dictated by the semantics

MR: $\text{bowner(player(our,2))}$

A semantic derivation:

```
  bowner(player(our,2))
   /     \
  /       \
bowner(_) player(our,2)
   /     \
  /       \
our  player(_,_) 2
```

```
  bowner(_) has null null
     /     \
    /       \
  the   ball
```
Relations to Other NLP Tasks: Syntactic Parsing

- Semantic parsing inherently includes syntactic parsing but as dictated by the semantics

MR: `bowner(player(our,2))`

A semantic derivation:

```
S-bowner(player(our,2))
```

```
NP-player(our,2)
```

```
PRP$-our NN-player(_,_) CD-2
```

```
our player 2
```

```
VP-bowner(_)
```

```
VB-bowner(_) null
```

```
has null null
```

```
the ball
```
Relations to Other NLP Tasks: Natural Language Generation

- Reversing a semantic parsing system becomes a natural language generation system [Jacobs, 1985; Wong & Mooney, 2007a]

Which rivers run through the states bordering Mississippi?

answer(traverse(next_to(stateid('mississippi'))))
Relations to Other NLP Tasks

• Tasks being performed within semantic parsing
  • Word sense disambiguation
  • Syntactic parsing as dictated by semantics

• Tasks closely related to semantic parsing
  • Machine translation
  • Natural language generation
Interpreting Language is Hard!

I saw a girl with a telescope

- “Parsing” resolves structural ambiguity in a formal way
Two Types of Parsing

- **Dependency**: focuses on relations between words
  
  ![Dependency Diagram](image)

- **Phrase structure**: focuses on identifying phrases and their recursive structure
  
  ![Phrase Structure Diagram](image)
Dependencies Also Resolve Ambiguity

I saw a girl with a telescope

I saw a girl with a telescope
Dependencies

- **Typed:** Label indicating relationship between words

  ![Typed Dependencies Diagram]

  *I saw a girl with a telescope*

- **Untyped:** Only which words depend

  ![Untyped Dependencies Diagram]

  *I saw a girl with a telescope*
Dependency Grammars

• In CFG-style phrase-structure grammars the main focus is on *constituents*.
• But it turns out you can get a lot done with just binary relations among the words in an utterance.
• In a dependency grammar framework, a parse is a tree where
  • the nodes stand for the words in an utterance
  • The links between the words represent dependency relations between pairs of words.
    • Relations may be typed (labeled), or not.
# Dependency Relations

<table>
<thead>
<tr>
<th>Argument Dependencies</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nsubj</td>
<td>nominal subject</td>
</tr>
<tr>
<td>csubj</td>
<td>clausal subject</td>
</tr>
<tr>
<td>dobj</td>
<td>direct object</td>
</tr>
<tr>
<td>iobj</td>
<td>indirect object</td>
</tr>
<tr>
<td>pobj</td>
<td>object of preposition</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Modifier Dependencies</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tmod</td>
<td>temporal modifier</td>
</tr>
<tr>
<td>appos</td>
<td>appositional modifier</td>
</tr>
<tr>
<td>det</td>
<td>determiner</td>
</tr>
<tr>
<td>prep</td>
<td>prepositional modifier</td>
</tr>
</tbody>
</table>
They hid the letter on the shelf
Dependency Parsing

• The dependency approach has a number of advantages over full phrase-structure parsing.
  • Deals well with free word order languages where the constituent structure is quite fluid
  • Parsing is much faster than CFG-bases parsers
  • Dependency structure often captures the syntactic relations needed by later applications
    • CFG-based approaches often extract this same information from trees anyway.
Dependency Parsing

• There are two modern approaches to dependency parsing
  • Optimization-based approaches that search a space of trees for the tree that *best* matches some criteria
  • Shift-reduce approaches that greedily take actions based on the current word and state.
Phrase Structure Tree

```
S
  /\  /
 NP VP
 /  /
 NP PP
 /  /
 JJ NNS IN DT NNS VBD VBG NNS
 Red figures on the screens indicated falling stocks
```
Dependency Grammar

- Syntactic structure consists of **lexical items**, linked by binary asymmetric relations called **dependencies**

- Interested in grammatical relations between individual words (**governing** & **dependent** words)

- Does not propose a recursive structure
  - Rather a network of relations

- These relations can also have labels
Draw the dependency tree

- Red figures on the screens indicated falling stocks
Dependency Tree

Red figures on the screens indicated falling stocks
Dependency Tree Example

- Phrasal nodes are missing in the dependency structure when compared to constituency structure.
Dependency Tree with Labels

Red figures on the screens indicated falling stocks
Comparison

• Dependency structures explicitly represent
  • Head-dependent relations (directed arcs)
  • Functional categories (arc labels)
  • Possibly some structural categories (parts-of-speech)

• Phrase structure explicitly represent
  • Phrases (non-terminal nodes)
  • Structural categories (non-terminal labels)
  • Possibly some functional categories (grammatical functions)
Learning DG over PSG

- Dependency Parsing is more straightforward
  - Parsing can be reduced to labeling each token \( w_i \) with \( w_j \)

- Direct encoding of predicate-argument structure
  - Fragments are directly interpretable

- Dependency structure independent of word order
  - Suitable for free word order languages (like Indian languages)
Dependency Tree

• Formal definition
  • An input word sequence \( w_1 \ldots w_n \)
  • Dependency graph \( D = (W,E) \) where
    • \( W \) is the set of nodes i.e. word tokens in the input seq.
    • \( E \) is the set of unlabeled tree edges \((w_i, w_j)\) \((w_i, w_j \in W)\).
    • \((w_i, w_j)\) indicates an edge from \( w_i \) (parent) to \( w_j \) (child).

• Task of mapping an input string to a dependency graph satisfying certain conditions is called dependency parsing.
Well-formedness

- A dependency graph is well-formed iff

  - **Single head**: Each word has only one head.
  
  - **Acyclic**: The graph should be acyclic.
  
  - **Connected**: The graph should be a single tree with all the words in the sentence.
  
  - **Projective**: If word A depends on word B, then all words between A and B are also subordinate to B (i.e. dominated by B).
Dependency Parsing

- Dependency based parsers can be broadly categorized into:
  - **Grammar driven** approaches
    - Parsing done using grammars.
  - **Data driven** approaches
    - Parsing by training on annotated/un-annotated data.
Dependency Parsing

• Dependency based parsers can be broadly categorized into
  • Grammar driven approaches
    • Parsing done using grammars.
  • Data driven approaches
    • Parsing by training on annotated/un-annotated data.

• These approaches are not mutually exclusive
Covington’s Incremental Algorithm

- Incremental parsing in $O(n^2)$ time by trying to link each new word to each preceding one [Covington 2001]:

\[
\text{PARSE}(x = (w_1, \ldots, w_n))
\]

1. \textbf{for} $i = 1$ \textbf{up to} $n$
2. \textbf{for} $j = i - 1$ \textbf{down to} $1$
3. \textbf{LINK}(w_i, w_j)
Covington’s Incremental Algorithm

- Incremental parsing in $O(n^2)$ time by trying to link each new word to each preceding one [Covington 2001]:

\[
\text{PARSE}(x = (w_1, \ldots, w_n))
\]

1. for $i = 1$ up to $n$
2. for $j = i - 1$ down to 1
3. LINK($w_i$, $w_j$)

- Different conditions, such as Single-Head and Projectivity, can be incorporated into the LINK operation.
Dynamic Programming

- Basic Idea: Treat dependencies as constituents.
- Use, e.g., CYK parser (with minor modifications)
Dynamic Programming Approaches

- Original version [Hays 1964] (grammar driven)
- Link grammar [Sleator and Temperley 1991] (grammar driven)
- Bilexical grammar [Eisner 1996] (data driven)
- Maximum spanning tree [McDonald 2006] (data driven)
Eisner 1996

- Two novel aspects:
  - Modified parsing algorithm
  - Probabilistic dependency parsing
- Time requirement: $O(n^3)$
- Modification: Instead of storing subtrees, store spans
- **Span**: Substring such that no interior word links to any word outside the span.
- **Idea**: In a span, only the boundary words are active, i.e. still need a head or a child
- One or both of the boundary words can be active
Example

Red figures on the screen indicated falling stocks
Example

Red figures on the screen indicated falling stocks

Spans:

\{ Red, figures \}  \{ indicated, falling, stocks \}
Assembly of correct parse

Start by combining adjacent words to minimal spans

{Red figures} {figures on} {on the}
Assembly of correct parse

Combine spans which overlap in one word; this word must be governed by a word in the left or right span.
Assembly of correct parse

Combine spans which overlap in one word; this word must be governed by a word in the left or right span.

\{ figures on \} + \{ on the screen \} \rightarrow \{ figures on the screen \}
Assembly of correct parse

Combine spans which overlap in one word; this word must be governed by a word in the left or right span.

Invalid span
Assembly of correct parse

Combine spans which overlap in one word; this word must be governed by a word in the left or right span.
Classifier-Based Parsing

• Data-driven deterministic parsing:
  • Deterministic parsing requires an oracle.
  • An oracle can be approximated by a classifier.
  • A classifier can be trained using treebank data.

• Learning algorithms:
  • Memory-based learning (MBL) [Nivre et al. 2004, Nivre and Scholz 2004]
  • Maximum entropy modeling (MaxEnt) [Cheng et al. 2005]
Feature Models

• Learning problem:
  • Approximate a function from parser states, represented by feature vectors to parser actions, given a training set of gold standard derivations.

• Typical features:
  • Tokens and POS tags of:
    • Target words
    • Linear context (neighbors in S and Q)
    • Structural context (parents, children, siblings in G)
  • Can not be used in dynamic programming algorithms.
Feature Models

Maximum Spanning Tree

- Each dependency is an edge in a directed graph
- Assign each edge a score (with machine learning)
- Keep the tree with the highest score

(Graph) saw girl
(a) i

(Scored Graph)
6 saw 4
-1 2
1 7
-2 5

(Dependency Tree)
6 saw 4
1
7

(Chu-Liu-Edmonds Algorithm)
Dependency Parsers for download

- MST parser by Ryan McDonald
- Malt parser by Joakim Nivre
- Stanford parser
Outline

• Dependency Parsing
  • Formal definition
  • Dynamic programming
  • Supervised Classification

• Semantic Role Labeling
  • Propbank
  • Automatic SRL
  • FrameNet

• Abstract Meaning Representation
What is PropBank: From Sentences to Propositions

Proposition: $\text{meet}(\text{Powell}, \text{Zhu Rongji})$

When Powell met Zhu Rongji on Thursday they discussed the return of the spy plane.

$\text{meet}(\text{Powell}, \text{Zhu}) \quad \text{discuss}([\text{Powell}, \text{Zhu}], \text{return}(X, \text{plane}))$
A TreeBanked Sentence

S
  /   
/     
NP  VP
  /   
/     
DT JJ NN VBD NP NNS VBG NN
   /     /     /
  The  Supreme Court gave states working Leeway

“Non Terminal” (130969 ARGs)
“Terminal” (4246 ARGs)
The Supreme Court states working Leeway/ARG0.
Core Arguments

- Arg0 = agent
- Arg1 = direct object / theme / patient
- Arg2 = indirect object / benefactive / instrument / attribute / end state
- Arg3 = start point / benefactive / instrument / attribute
- Arg4 = end point
Secondary ArgMs

- LOC - where at?
- DIR - where to?
- MNR - how?
- PRP - why?
- REC - himself, themselves, each other
- PRD - this argument refers to or modifies another
- ADV – others
- TMP - when?
- TPC – topic
- ADV – others
Distributions of Argument Types

CORE Types
- ARG0: 91519
- ARG1: 26308
- ARG2: 11120
- ARG3: 26308
- ARG4: 74
- ARG5: 10100
- DIR: 11670
- LOC: 4502
- EXT: 2553
- REC: 2837
- TMP: 5600

ARGM Types
- ARG0: 14187
- ARG1: 8876
- ARG2: 18703
- ARG3: 116628
- ARG4: 21670
- ARG5: 8876
- DIR: 11670
- LOC: 4502
- EXT: 2553
- REC: 2837
- TMP: 5600
How to Use PropBank: Train a Semantic Role Labeling System

- (CONLL 04, CONLL 05 and) Our Goal: Given a list of (3073) target verbs the system should be able to tag the possible nodes with semantic role labels (Ji et al., 2005)
Predicate Features: Lexical

• Head Word, Head Pos of (-2,-1,0,1,2) window of Predicate

• Predicate is a Transitive verb or not

• Predicate Voice (Passive or not)
  • Verb itself: must be in its past particle form
  • Passive Context
    --Immediately following the verb "be"
    --Postmodifying a noun in a reduced relative clause, "The building damaged by fire".

• Encoding
  Conjunction feature of Predicate POS_Passive Context
Predicate Sub-Categorization Feature

The Phrase structure rule expanding the predicate’s grandparent

The Phrase structure rule expanding the predicate’s grandparent
Predicate Pivot Features

Consider the predicate as a “Pivot”, and its grandparent’s children are defined in relation to it.
Argument Features: Lexical/Syntactic

- Head Word, Head Pos, Phrase Type of (-2,-1,0,1,2) window words, Begin Word, Last Word, Left Sister, Right Sister, Parent of Argument
  (Head of PP replaced by head word of NP inside it)
- Head Pos, Phrase Type of GrandParent
- Suffix1, Suffix2, Suffix3
- Preceding, Succeeding Node’s Label
- Length (Span)
- Level from Leaves
- Beginning Letter of Phrase Type (for generalization)
- Punctuation before/after
- If it includes a Preposition or not, and Prep POS
Intervene Feature: Path

Path: NP ↑ S ↓ VP ↓ VBD ↓ gave
PartialPath: NP ↑ S ↓
InterPath: S ↑ VP ↓ VBD
PredPathArgPathLens: 3_1
CollapsedPath (delete nodes between clauses): NP ↑ S ↓ VP ↓ VBD ↓ gave
PathSOnly (Replace all the non-clause nodes with "*"): NP ↑ S ↓ * ↓ * ↓ gave
PathSRemain (only keep clause nodes): NP ↑ S ↓ gave
PathNGrams: NP|S|VP, S|VP|VBD, VP|VBD|gave, VBD|gave|*, gave|*|*
PathLen: 4
ArgPhraseType, PathLen: NP, 4
PredHeadWord, PathSRemain: gave, NP ↑ S ↓ gave
Intervene Feature: Position

```
NP/ARG0
DTJJNN
The  Supreme Court

VBD
...  gave

Directionality: Left     SameClause: true     Dominate Phrase Type: S
Adjacency (Adjacent/not-Adjacent): Adjacent
ArgPhraseType,Adjacency: NP,Adjacent
PositionInClause (Begin, Inside, End): Begin
RelPosition (Distance between Spans): 0
RelPosition,Directionality: 0, Left     RelPosition,Transitive: 0, True
RelPosition,PredHeadPos,PassiveContext: 0, VBD, False
RelPosition,ArgPhraseType: 0, NP
RelPosition,PredPivotv: 0, v_NNS_VBG_NN
```
Intervene Features: Pivot

Consider the predicate and candidate argument as “Pivots”, and other constituents are defined in relation to them.
Other Features

- PredHeadWord, ArgHeadWord
- PredHeadWord, ArgPhraseType
- ArgPreposition, Transitive
- Frequency of VP, NP, SBAR, CC, “,”, “:”, “”” in the sentence
- …
- …
Website Back Up :-)  

The scheduled maintenance was completed on Saturday, May 17th, and this website should now be up and fully functional. We have noticed that the startup of the top page is sometimes very slow, although the rest of the site seems to function normally after that. We are looking into the cause of this problem. If you notice any other problems with the website, or evidence of misuse, Please contact us. Thank you for your patience.

Last Updated (May 20, 2008 at 11:30 AM)

Welcome to FrameNet  

The Berkeley FrameNet project is creating an on-line lexical resource for English, based on frame semantics and supported by corpus evidence. The aim is to document the range of semantic and syntactic combinatory possibilities (valences) of each word in each of its senses, through computer-assisted annotation of example sentences and automatic tabulation and display of the annotation results. The major product of this work, the FrameNet lexical database, currently contains more than 10,000 lexical units (defined below). more than 6,100 of which

The "Book" v1.3 Available for Download  

Our main project document, "FrameNet II: Extended Theory and Practice" gives a basic introduction to frame semantics and offers guidelines for frame semantic annotation, discussing in detail the inventories of grammatical functions and phrase types that we use. The document includes a discussion of the frame development process and a consolidated list of frame-general, extra-thematic frame elements. Our latest version contains some general updates plus a new section
Hmm. Haven’t I heard that word “frame” before?

Yes, it’s intended to be seen as a variation on the word as it’s been used in various branches of the cognitive sciences in recent decades.
“Frames” Traditions

Let’s locate “our” notion of frame within the various traditions in the cognitive sciences that use the words frame or schema (joined, sometimes, with stereotype, script, scenario or idealized cognitive model) which appear to be dealing with essentially the same concept.

These terms are used to denote structured sets of expectations that play a central role in how human beings create or interpret their experiences.
customer

The noun customer is typically defined as ‘someone who buys something in a shop or business.’ That includes everyone I know over the age of 5.

Suppose you overhear somebody say

Sue tends to be rude to customers.

What situation do you imagine?
chicken (*mass noun*)

- The noun **chicken**, as a *count noun*, is the name of a well-known domestic bird. As a mass noun it is defined as ‘the meat of a chicken or chickens’.
- What’s wrong with the following sentence?
  
  The fox that lives near our farm **likes chicken**. *(compare: likes chickens)*

- The image you might get is of a fox eating fried chicken, holding a knife and a fork, and a napkin, in its paws.
• The products of the lexical construction that yields mass noun uses of *chicken, lamb, duck, turkey*, etc., refer to *meats prepared as part of human cuisine*.

*The wolf that lives near our ranch prefers lamb.*
Invoking and Evoking Frames

- **People invoke** (summon up from their memory) frames, to make sense of their experience, linguistic or otherwise.
  - a cognitive act

- **Words evoke** categories and knowledge structures that shape interpreters’ understanding of a text.
  - a cognitive experience

*Warning: this not a standard use of these words.*
So,

We need to describe words in terms of the “franal” background. If we don’t understand the frame, we don’t understand the word, or why the language needs this word, or why the speaker chose to use it.
The ideal dictionary should let you

1. Look up a word
2. Get a link to a description of the relevant frame, for each of its meanings, and see the names of the frame’s components
3. See a display of its combinatory affordances, its **valence** possibilities, both semantic and syntactic
4. Find a collection of example sentences illustrating all of its main combinatory patterns
5. Find a list of other words that evoke the same frame
6. Link to other semantically related frames
Frame examples: Risk

Taking_a_risk:
- Protagonist, Action, Harm, Asset
  1. I’m going to risk a swim in the sea.
  2. You’ll risk bankruptcy if you make that investment.
  3. You’re risking your reputation by doing that.
  4. You’re taking a big risk.

Being_at_risk:
- Protagonist, Harm, Asset
  2. Newborns in this hospital run the risk of hypothermia.
  3. We risk our lives every day.
  4. I am at risk of stroke.
Frame examples: Explanation

Communication.explanation:
   Speaker, Addressee, Mystery, Account
1. The coach explained the problem to the team.
2. The coach explained that they hadn't learned the maneuvers.
3. What's your explanation of these facts?
4. The defense lawyer gave an inadequate explanation.

Cognition.explanation:
   Mystery, Account
1. What can explain these facts?
2. A history of unrestricted logging explains the erosion pattern we see here.
Compare: explain & reveal

In their cognitive senses, as opposed to their meanings as verbs of speaking, the verbs explain and reveal are near-inverses of each other, where the Mystery and the Account in the former correspond to Evidence and Conclusion in the latter.

1. A history of unrestrained logging explains the erosion pattern we see here. (explains, accounts for)
2. The erosion pattern we see here reveals a history of unrestrained logging. (reveals, shows, suggests)
# Reading the Pictures

- The boxes refer to five-part scenarios consisting of an initial state, a transition, an intermediary state, another transition, and a final state.
- The writing under the pictures abbreviates particular role names and gives verbs that evoke instances of the scenario.
- The bold borders indicate a profiling of some portion of the event.

<table>
<thead>
<tr>
<th>state</th>
<th>transition</th>
<th>state</th>
<th>transition</th>
<th>state</th>
</tr>
</thead>
</table>


X; return, go back, come back

He returned to Hong Kong.
He returned Tuesday evening after a week’s trip to Australia.
He returned to his home for a few days.

*The verb RETURN profiles the time of arrival, but it evokes the entire frame; other information in the sentences can fill in some of the details of the larger scenario.*
I returned your books this morning.

I returned to your desk the books that I had borrowed last week.

After the earthquake we replaced all the books on the shelf.
frame elements

• Participants and Sub-events.
  • Avenger the one who enacts revenge
  • Offender the original offender
  • Injured_party the offender’s victim
  • Injury the offender’s act
  • Punishment the avenger’s act
grammar

• Components of linguistic form for expressing the FEs (defining **valence**).
  • **Subject**
  • **Direct Object**
  • **Prepositional marking**
    (*by, for, with, on, at, against*)
  • **Subordinate clause marking**
    (*for DOING, by DOING*)
Outline

• Dependency Parsing
  • Formal definition
  • Dynamic programming
  • Supervised Classification

• Semantic Role Labeling
  • Propbank
  • Automatic SRL
  • FrameNet

• Abstract Meaning Representation
Why does AMR matter now?

• AMR is a semantic representation aimed at large-scale human annotation in order to build a giant semantics bank.

• We do a practical, replicable amount of abstraction (limited canonicalization).

• Capture many aspects of meaning in a single simple data structure.

Hasn’t this been done before?

• Linguistics/CL have formalized semantics for a long time.

• A form of AMR has been around for a long time too (Langkilde and Knight 1998).

• It changed a lot since 1998 (add PropBank, etc.) and we actually built a corpus of AMRs.
Contemporary AMR

- Banarescu et al. (2013) laid out the fundamentals of the annotation scheme we’ll describe today.

Roadmap for Part I

- Fundamentals of the AMR representation
- Hands-on practice I: Representing basic examples
- Advanced topics and linguistic phenomena
- Comparison to other representations
- Hands-on practice II: Doing real, complex text
• We use PENMAN notation (Bateman 1990).

• A way of representing a directed graph in a simple, tree-like form.

“The dog is eating a bone”

(e / eat-01
 :ARG0 (d / dog)
 :ARG1 (b / bone))

• The edges (ARG0 and ARG1) are relations

• Each node in the graph has a variable

• They are labeled with concepts

• d / dog means “d is an instance of dog”

“The dog is eating a bone”

(e / eat-01
 :ARG0 (d / dog)
 :ARG1 (b / bone))
**PENMAN notation**

- **Concepts** are technically *edges* (this matters in Part 2)

“*The dog is eating a bone*”

\[
\text{eat-01} : \text{ARG0} (d / \text{dog}) \\
\text{ARG1} (\text{b} / \text{bone})
\]

---

**Reentrancy**

- What if something is referenced multiple times?

- Notice how *dog* has two incoming roles now.

- To do this in PENMAN format, repeat the variable. We call this a **reentrancy**.

“*The dog wants to eat the bone*”

\[
\text{want-01} : \text{ARG0} (d / \text{dog}) \\
\text{ARG1} (\text{e} / \text{eat-01}) \\
\text{ARG0} (\text{d} / \text{dog}) \\
\text{ARG1} (\text{b} / \text{bone})
\]
• It **does not matter** where the concept label goes.

“The dog wants to eat the bone”

(want-01
 :ARG0 (d / dog)
 :ARG1 (e / eat-01
 :ARG0 d
 :ARG1 (b / bone)))

Inverse Relations and Focus

• What about “The dog ate the bone that he found”?

• How do we know what goes on top?

• How do we get these into the AMR format?
Inverse Relations and Focus

• We call “what goes on top” the focus.
• Conceptually, the main assertion.
• Linguistically, often the head.
  ▸ For a sentence, usually the main verb.
Inverse Relations and Focus

The man at the hotel

\[(m / \text{man} : \text{location} (h / \text{hotel}))\]

The dog ran

\[(r / \text{ran-01} : \text{ARG0} (d / \text{dog}))\]

Inverse Relations and Focus

The hotel the man is at

\[(h / \text{hotel} : ??? (m / \text{man}))\]

The dog that ran

\[(d / \text{dog} : ????? (r / \text{ran-01}))\]
Inverse Relations and Focus

- This is a notational trick: \( X \text{ ARG0-of } Y = Y \text{ ARG0 X} \)

- Often used for relative clauses.

- *These are equivalent for SMATCH scoring purposes too.*
Reviewing the Format

• Imagine a graph for “The dog ate the bone that he found”

• “The dog ate the bone that he found”

Reviewing the Format

• Find the focus

• “The dog ate the bone that he found”
Reviewing the Format

• Add entities

Focus ➔ eat-01 ➔ ARG0 ➔ dog ➔ ARG0 ➔ find-01 ➔ ARG1 ➔ bone ➔ ARG1

• “The dog ate the bone that he found”

Reviewing the Format

• Invert a relation if needed

Focus ➔ eat-01 ➔ ARG0 ➔ dog ➔ ARG0 ➔ find-01 ➔ ARG1 ➔ bone ➔ ARG1-of

• “The dog ate the bone that he found”
Reviewing the Format

• Add reentrancies

Focus eat-01 ARG0 dog ARG1 bone ARG1-of find-01

• “The dog ate the bone that he found”

Constant

• Some relations, called constants, get no variable.

• The editor does this automatically for certain contexts.

• This happens for negation.

“The dog did not eat the bone”
(e / eat-01 :polarity -
 :ARG0 (d / dog)
 :ARG1 (b / bone))
• Some relations, called constants, get no variable.

• The editor does this automatically for certain contexts.

• This happens for numbers.

“\textit{The dog ate four bones}”

\begin{verbatim}
(e / eat-01
 :ARG0 (d / dog)
 :ARG1 (b / bone :quant 4))
\end{verbatim}

\textit{(to create a concept starting with a nonalphabetic character, type “!” before the concept)}

• Some relations, called \textbf{constants}, get no \textbf{variable}.

• The editor does this \textbf{automatically} for certain contexts.

• This happens for \textbf{names}.

“\textit{Fido the dog}”

\begin{verbatim}
(d / dog
 :name (n / name :op1 "Fido"))
\end{verbatim}
Concepts vs. Constants

• A **concept** is a type. For every concept node there will be $\geq 1$ instance variable/node.

• An instance can be mentioned multiple times.

• Multiple instances of the same concept can be mentioned.

• **Constants** are singleton nodes: no variable, just a value. Specific non-core roles allow constant values.

• That’s AMR notation! Let’s review before discussing how we annotate AMRs.

```
(e / eat-01
 :ARG0 (d / dog)
 :ARG1 (b / bone
  :quant 4
  :ARG1-of (f / find-01
   :ARG0 d)))))
```

variable  concept  constant

inverse relation  reentrancy
PropBank Lexicon

• Predicates use the PropBank inventory.

• Each frame presents annotators with a list of senses.

• Each sense has its own definitions for its numbered (core) arguments.

PropBank Lexicon

• We generalize across parts of speech and etymologically related words:

  My fear of snakes  fear-01
  I am fearful of snakes  fear-01
  I fear snakes  fear-01
  I’m afraid of snakes  fear-01

• But we don’t generalize over synonyms:

  My fear of snakes  fear-01
  I’m terrified of snakes  terrify-01
  Snakes creep me out  creep_out-03
Stemming Concepts

• Non-predicates don’t have PropBank frames. They are simply stemmed.

• All concepts drop **plurality**, **articles**, and **tense**.

<table>
<thead>
<tr>
<th>A cat</th>
<th>eating</th>
</tr>
</thead>
<tbody>
<tr>
<td>The cat</td>
<td>eats</td>
</tr>
<tr>
<td>cats</td>
<td>ate</td>
</tr>
<tr>
<td>the cats</td>
<td>will eat</td>
</tr>
<tr>
<td>(c / cat)</td>
<td>(e / eat-01)</td>
</tr>
</tbody>
</table>

Why drop articles?

• All mentions of a term go to **the same variable**, including **pronouns** and **later nominal mentions**.

* I saw a **nice** dog and noticed he was eating a bone

Is “d” indefinite or definite?

• We do capture **demonstratives**:

* This house (h / house :mod (t / this))
Stemming Concepts

- Pronouns that do not have a coreferent nominal mention are made nominative and used as normal concepts.

The man saved himself  He saved himself  He saved me

(s / save-01   (s / save-01   (s / save-01
  :ARG0 (m / man) :ARG0 (h / he) :ARG0 (h / he)
  :ARG1 m) :ARG1 h) :ARG1 (i / i))

Why drop tense?

- English verbal tense doesn’t generalize well cross-linguistically; not available for nominal predicates.

- Richer time representation might have required looking beyond a sentence.

- Keep a simple representation.
The man described the mission as a disaster.
The man's description of the mission: disaster.
As the man described it, the mission was a disaster.
The man described the mission as disastrous.

(d / describe-01
  :ARG0 (m / man)
  :ARG1 (m2 / mission)
  :ARG2 (d / disaster))

Non-core Role Inventory

- If a semantic role is not in the **core roles** for a roleset, AMR provides an inventory of **non-core roles**
- These express things like **:time, :manner, :part, :location, :frequency**
- Inventory on handout, or in editor (the [roles] button)
Non-core Role Inventory

- We use :mod for attribution, and :domain is the inverse of mod (:domain = :mod-of)

\[
\begin{align*}
\text{The yummy food} & \quad \text{The yumminess of the food} \\
\text{There is yummy food} & \quad \text{The food is yummy}
\end{align*}
\]

\[
\begin{align*}
\text{(f / food} & \quad \text{(y / yummy)} \\
\text{:mod (y / yummy))} & \quad \text{:domain (f / food))}
\end{align*}
\]

- This is also used for attribute/predicative demonstratives and nominals

\[
\begin{align*}
\text{This house} & \quad \text{This is a house} \\
\text{(h / house} & \quad \text{(t / this)} \\
\text{:mod (t / this))} & \quad \text{:domain (h / house))}
\end{align*}
\]

\[
\begin{align*}
\text{A monster truck} & \quad \text{the truck is a monster} \\
\text{(t / truck} & \quad \text{(m / monster)} \\
\text{:mod (m / monster))} & \quad \text{:domain (t / truck))}
\end{align*}
\]
Non-core Roles: \texttt{op}#

- Some relations need to have an ordered list of arguments, but \textbf{don’t have specific meanings} for each entry.

- We use \texttt{op1}, \texttt{op2}, \texttt{op3}, … for these

\texttt{op}# for coordination

- We use this for coordination:

- \textit{Apples and bananas} (a / and \texttt{op1} (a2 / apple) \texttt{op2} (b / banana))
:op# for names

• Barack Obama
  (p / person
   :name (n / name
     :op1 "Barack"
     :op2 "Obama"))

• Obama
  (p / person
   :name (n / name
     :op1 "Obama"))

Named Entities

• Barack Obama
  (p / person
   :name (n / name
     :op1 "Barack"
     :op2 "Obama"))

• Entities with names get special treatment!

• We assign a named entity type from our ontology.

• 70+ categories like person, criminal-organization, newspaper, city, food-dish, conference

• See your handout, or the [NE types] button in the editor
Named Entities

- *Barack Obama* (p / person :name (n / name :op1 "Barack" :op2 "Obama"))

- Entities with names get special treatment!

- Each gets a :name relation to a name node

- That node gets :op# relations to the strings of their name *as used in the sentence.*

Named Entities

- If there is a more specific descriptor present in the sentence, we use that instead of the NE inventory.

- *a Kleenex* (p / product :name (n / name :op1 "Kleenex"))

- *a Kleenex tissue* (t / tissue :name (n / name :op1 "Kleenex"))
Wikification

• In a second pass of annotation, we add :wiki relations.

• Barack Obama

http://en.wikipedia.org/wiki/Barack_Obama

Measurable Entities

• We also have special entity types we use for normalizable entities.

“Tuesday the 19th”

(d / date-entity
 :weekday (t / tuesday)
 :day 19)

“five bucks”

(m / monetary-quantity
 :unit dollar
 :quant 5)
Measurable Entities

• We also have special entity types we use for normalizable entities.

“$3 / gallon”

(r / rate-entity-91
 :ARG1 (m / monetary-quantity
   :unit dollar
   :quant 3)
 :ARG2 (v / volume-quantity
   :unit gallon
   :quant 1))