Supplementary Material: Broad-coverage CCG Semantic Parsing with AMR

Yoav Artzi[*]
Dept. of Computer Science and Cornell Tech
Cornell University
New York, NY 10011
yoav@cs.cornell.edu

Kenton Lee **Luke Zettlemoyer**
Computer Science & Engineering
University of Washington
Seattle, WA 98195
{kentonl, lsz}@cs.washington.edu

1 Features

For feature computation, we append to each binary predicate originating from a single-token lexical entry the token that triggered it. For example, given the lexical entry *dance* $\vdash \lambda x.\lambda d.\text{dance-01}(d) \wedge \text{ARG0}(d, x)$, when computing features, we will consider its logical form to be $\lambda x.\lambda d.\text{dance-01}(d) \wedge \text{ARG0+}\mathit{dance}(d, x)$. For simplicity, where possible, we abstract this step in this section. We also simplify feature names.

1.1 CCG Parsing Features

Lexical Features We create multiple features for each lexical entry:

- Indicator features for every lexeme, lexical template and every pairing of lexeme and template. For example, consider the lexical entry *dance* $\vdash \lambda x.\lambda d.\text{dance-01}(d) \wedge \text{ARG0}(d, x)$ generated from the lexeme $\langle \mathit{dance}, \{\text{dance-01}\}\rangle$ and the template $\lambda v_1.[S\backslash NP : \lambda x.\lambda a.v_1(a) \wedge \text{ARG0}(a, x)]$. Using this entry in a derivation will trigger three features: one for the lexeme, one for the template and one for pairing the two.

- Features to indicate using a lexical entry generated from the date and number regular expression heuristics. For example, the entry *2001* $\vdash NP : \mathcal{A}_1(\lambda d.\text{date-entity}(d) \wedge \text{year}(d, 2001))$ may be created from the date regular expression and will trigger the feature DATEREGEXP.

- For each usage of a lexical entry with a single token, we trigger a feature that pairs the part-of-speech tag of the token with its syntactic attribute. For example, if the word *weapons* was assigned the tag NNS and the lexical entry *weapons* $\vdash N_{[pl]} : \lambda w.\text{weapon}(w)$, we will create a feature NNS+pl.

[*] Work done at the University of Washington.

Logical Form Features We trigger an indicator feature if the logical expression at the root of a complete parse tree includes repeating elements in a conjunction. We ignore Skolem IDs when computing this feature. For example, the logical form

$$\begin{aligned}&\mathcal{A}_1(\lambda i.\text{involve-01}(i) \wedge\\ &\quad \text{ARG0}(i, \mathcal{A}_2(\lambda b.\text{boy}(b))) \wedge\\ &\quad \text{ARG0}(i, \mathcal{A}_3(\lambda b.\text{boy}(b))))\end{aligned}$$

will trigger this feature.

Parsing Operations We create a feature with the rule name for every use of a unary shifting rule and a feature that pairs the rule name with the instance typing predicate of the top instance in the logical form. For example, the unary parse step

$$\dfrac{\begin{array}{c}NP_{[sg]}\\ \mathcal{A}_1(\lambda c.\text{city}(c) \wedge \text{name}(c, \mathcal{A}_2(\lambda n.\text{name}(n) \wedge\\ \text{op}(n, \text{PYONGYANG}))))\end{array}}{\begin{array}{c}N_{[x]}/N_{[x]}\\ \lambda f.\lambda x.f(x) \wedge \text{REL}(x, \mathcal{A}_1(\lambda c.\text{city}(c) \wedge \text{name}(c,\\ \mathcal{A}_2(\lambda n.\text{name}(n) \wedge \text{op}(n, \text{PYONGYANG})))))\end{array}}$$

uses the rule ADJNP to shift a NP to an adjective and will trigger two features: ADJNP and ADJNP+city. We also create features for using forward or backward crossing composition.

Attachment Features When a parsing operation creates a new fully specified AMR relation, we create features using the instance type of both concepts and the relation name. For example, the forward application parsing step

$$\dfrac{\begin{array}{c}S\backslash NP/NP\\ \lambda x.\lambda y.\lambda d.\text{deny-01}(d) \wedge\\ \text{ARG0}(d, y) \wedge\\ \text{ARG1}(d, x)\end{array} \qquad \begin{array}{c}NP_{[nb]}\\ \mathcal{A}_7(\lambda i.\text{involve-01}(i) \wedge\\ \text{ARG1}(i, \mathcal{R}(\text{ID})))\\ \\ \end{array}}{\begin{array}{c}S\backslash NP\\ \lambda y.\lambda d.\text{deny-01}(d) \wedge \text{ARG0}(d, y) \wedge \text{ARG1}(d,\\ \mathcal{A}_7(\lambda i.\text{involve-01}(i) \wedge \text{ARG1}(i, \mathcal{R}(\text{ID}))))\end{array}}>$$

will create the features:

- deny-01+ARG1+involve-01
- deny-01+ARG1
- ARG1+involve-01

1.2 Constant Mapping Features

Unary Features For every constant assignment we create a simple bias feature that includes the assigned constant and a feature that pairs the assigned constant with the original constant in the underspecified logical form (including the appended origin token, if exists). For example, consider the phrase *king of thieves* that may be represented by the underspecified logical form (with appended tokens) $\mathcal{A}_1(\lambda k.\text{king}(k) \wedge \text{REL+}of(c, \mathcal{A}_2(\lambda t.\text{thief}(t))))$ and the fully specified $\mathcal{A}_1(\lambda k.\text{king}(k) \wedge \text{mod}(c, \mathcal{A}_2(\lambda t.\text{thief}(t))))$. This mapping will create the features mod and $\text{mod+REL+}of$.

Selectional Preferences For every relation in the logical form, we create features that include the typing predicate of each instance and the relation predicate. For example, for the mapping above, we will create the following three feature:

- king+mod+thief
- king+mod
- mod+thief

If one of the instances is a reference to a Skolem ID (e.g., $\mathcal{R}(2)$ in $\mathcal{A}_1(\lambda x.\text{love-01}(x) \wedge \text{ARG0}(x, \mathcal{A}_2(\lambda y.\text{boy}(y))) \wedge \text{ARG1}(x, \mathcal{R}(2))))$ we use the type of the referenced instance (boy, in the example).

Control Preferences We create features that account for references via control structures. For example, consider the sentence *I want to buy a ticket* that may be represented by the underspecified logical form

$$\begin{aligned}&\mathcal{A}_1(\lambda w.\text{want-01}(w) \wedge\\ &\quad \text{ARG0}(w, \mathcal{A}_2(\lambda i.\text{I}(i))) \wedge\\ &\quad \text{ARG1}(w, \mathcal{A}_3(\lambda b.\text{buy-01}(b) \wedge\\ &\qquad \text{ARG0}(b, \mathcal{R}(\text{ID})) \wedge\\ &\qquad \text{ARG1}(b, \mathcal{A}_4(\lambda t.\text{ticket}(t))))))\end{aligned}$$

and mapped to the fully specified

$$\begin{aligned}&\mathcal{A}_1(\lambda w.\textbf{want-01}(w) \wedge\\ &\quad \textbf{ARG0}(w, \mathcal{A}_2(\lambda i.\text{I}(i))) \wedge\\ &\quad \textbf{ARG1}(w, \mathcal{A}_3(\lambda b.\text{buy-01}(b) \wedge\\ &\qquad \textbf{ARG0}(b, \mathcal{R}(2)) \wedge\\ &\qquad \text{ARG1}(b, \mathcal{A}_4(\lambda t.\text{ticket}(t)))))) \ .\end{aligned}$$

This mapping will create the feature $\text{want-01+ARG0+ARG1+ARG0}$ for resolving ID to 2. The above fully specified form highlights the constants used in the feature.

2 AMR to Lambda Calculus Conversion

We define a deterministic and invertible conversion process between AMR and lambda-calculus representations. Table 1 describes one direction (AMR to lambda calculus) of the recursive CONVERT process. In practice, we implemented both directions, and are able to accurately convert between the representations. We define *instance type* following the AMR specification,[1] for example location, person, name, etc. CONVERT distinguishes between name instance types and all other instance types. For name, we concatenate the name tokens. For all other relations, we simply create conjunctions between binary literals and recursively convert the related instance. For a variable re-used for reference, we replace it with a reference to the appropriate Skolem ID. Every instance is assigned a unique Skolem ID. We create i-typed constants for numbers, txt-typed constants for concatenated names and e-typed constants for all other symbols.

Our simple and deterministic conversion process makes no distinction between references that are the result of syntactic control and non-compositional references (e.g., pronouns). For example, the structures for *Pyongyang officials denied their involvement* and *Pyongyang officials denied involvement* will be identical, although their syntactic derivation is different. As a result, our model, inference procedure and learned grammar do not make this linguistic distinction. While a more linguistically motivated approach to control structures is likely to increase the accuracy of resolving such dependencies, it will require treating the distinction between such cases and other references as latent variables, further complicating the learning process. We, therefore, consider this a promising direction for future work.

3 Example Derivations

Figures 1 and 2 show example output derivations from out system for two sentences from the development set. Each figure includes the max-scoring derivation, the AMR output of our learned model and the annotated AMR. We abstract over the details of the constant mapping step and represent it as the last step of the CCG parse tree.

[1] `https://github.com/amrisi/amr-guidelines/blob/master/amr.md`

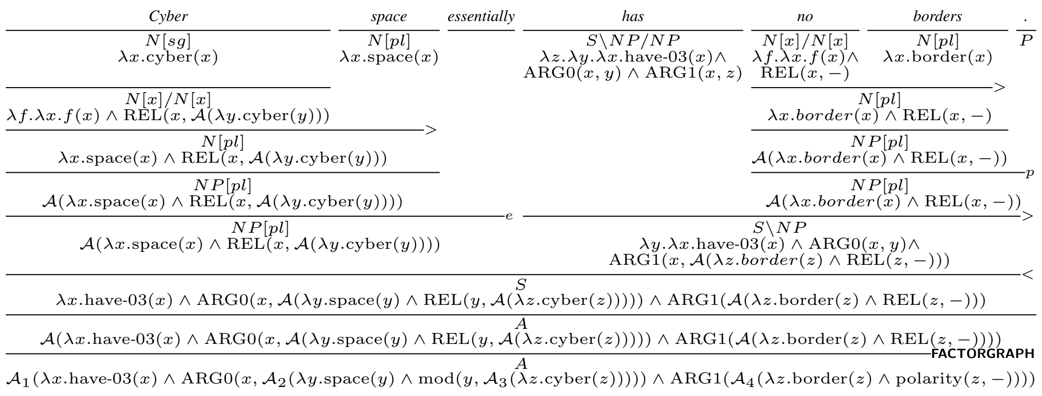
AMR A	Lambda calculus: $\textsc{Convert}(A)$	Update $\textsc{IdMap}$
Name instances		
$(n/\text{name}$	$\mathcal{A}_k(\lambda n.\text{name}(n) \wedge$	$\textsc{IdMap}[n] = k$
:op1 TOKEN1	$\text{op}(t, TOKEN1\_TOKEN2\_$	
:op2 TOKEN2	$\ldots\_TOKENm) \wedge$	
$\ldots$	$\text{relation1}(t, \textsc{Convert}(A1)) \wedge$	
:opm TOKENm	$\text{relation2}(t, \textsc{Convert}(A2)) \wedge$	
:relation1 $A1$	$\cdots \wedge$	
:relation2 $A2$	$\text{relationl}(t, \textsc{Convert}(l))$	
$\ldots$		
:relationl $Al)$		
All other instances		
$(t/\text{instance\_type}$	$\mathcal{A}_k(\lambda t.\text{instance\_type}(t) \wedge$	$\textsc{IdMap}[t] = k$
:relation1 $A1$	$\text{relation1}(t, \textsc{Convert}(A1)) \wedge$	
:relation2 $A2$	$\text{relation2}(t, \textsc{Convert}(A2)) \wedge$	
$\ldots$	$\cdots \wedge$	
:relationl $Al)$	$\text{relationl}(t, \textsc{Convert}(Al))$	
Re-using a variable for reference		
v	$\mathcal{R}(\textsc{IdMap}[v])$	
Other AMR symbols		
symbol	symbol	

Table 1: AMR to lambda calculus conversion cases. For clarity, we omit the typing information. IDMAP maintains a mapping of variables to Skolem IDs.

4 Comparisons to JAMR

We provide comparisons between our system outputs and JAMR outputs for sentences from the development set. Figures 3 through 9 show for each example the annotated AMR, the output of our approach and the JAMR output.

As expected, JAMR's weaker model of syntax results in erroneous distant attachments that violate the syntactic structure of the sentence. In contrast, while our approach also suffers from attachment errors, in general, they are more similar to these seen in syntactic parsers and are derived from plausible syntactic decisions. As observed in the overall results, we also see in the examples below that JAMR displays significantly lower recall. While JAMR is limited to surface-form heuristics only, our approach adopts, in addition, a more liberal approach to generate lexical entries and is able to entertain more high quality hypotheses due to the usage of syntactic constraints. This allows our approach to recover more complete sentential representations. Finally, we observe that our approach generates more human readable derivations. We are able to easily identify the origin of attachment decisions by observing the lexical entries assigned to words.



Cyber
space
essentially
has
no
borders
.
N[sg] λx.cyber(x)
N[pl] λx.space(x)
S\NP/NP λz.λy.λx.have-03(x)∧ ARG0(x, y) ∧ ARG1(x, z)
N[x]/N[x] λf.λx.f(x)∧ REL(x, −)
N[pl] λx.border(x)
P
N[x]/N[x] λf.λx.f(x) ∧ REL(x, A(λy.cyber(y)))
N[pl] λx.space(x) ∧ REL(x, A(λy.cyber(y)))
NP[pl] A(λx.space(x) ∧ REL(x, A(λy.cyber(y))))
NP[pl] A(λx.space(x) ∧ REL(x, A(λy.cyber(y))))
N[pl] λx.border(x) ∧ REL(x, −)
NP[pl] A(λx.border(x) ∧ REL(x, −))
NP[pl] A(λx.border(x) ∧ REL(x, −))
S\NP λy.λx.have-03(x) ∧ ARG0(x, y)∧ ARG1(x, A(λz.border(z) ∧ REL(z, −)))
S λx.have-03(x) ∧ ARG0(x, A(λy.space(y) ∧ REL(y, A(λz.cyber(z))))) ∧ ARG1(A(λz.border(z) ∧ REL(z, −)))
A A(λx.have-03(x) ∧ ARG0(x, A(λy.space(y) ∧ REL(y, A(λz.cyber(z))))) ∧ ARG1(A(λz.border(z) ∧ REL(z, −))))
FACTORGRAPH
A A1(λx.have-03(x) ∧ ARG0(x, A2(λy.space(y) ∧ mod(y, A3(λz.cyber(z))))) ∧ ARG1(A4(λz.border(z) ∧ polarity(z, −))))

(a) Derivation of the final logical form.

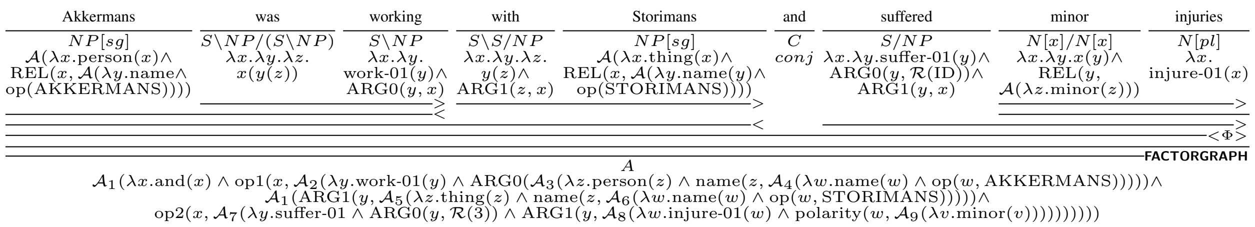
```
(h/have-03
  :ARG0 (s/space
    :mod (c/cyber))
  :ARG1 (b/border
    :polarity −)
```

(b) Predicted AMR.

```
(h/have-03
  :ARG0 (s/space
    :mod (c/cyber))
  :ARG1 (b/border)
  :polarity −
  :manner (e/essential))
```

(c) Annotated AMR.

Figure 1: The max-scoring derivation for the sentence *Cyber space essentially has no borders*. The system failed to recover the manner relation of have-03, and it incorrectly attached the negation to border rather than to have-03. The SMATCH F1 for this example is 0.80.



Akkermans
was
working
with
Storimans
and
suffered
minor
injuries
NP[sg] A(λx.person(x)∧ REL(x, A(λy.name∧ op(AKKERMANS))))
S\NP/(S\NP) λx.λy.λz. x(y(z))
S\NP λx.λy. work-01(y)∧ ARG0(y, x)
S\S/NP λx.λy.λz. y(z)∧ ARG1(z, x)
NP[sg] A(λx.thing(x)∧ REL(x, A(λy.name(y)∧ op(STORIMANS))))
C conj
S/NP λx.λy.suffer-01(y)∧ ARG0(y, R(ID))∧ ARG1(y, x)
N[x]/N[x] λx.λy.x(y)∧ REL(y, A(λz.minor(z)))
N[pl] λx. injure-01(x)
FACTORGRAPH
A A1(λx.and(x) ∧ op1(x, A2(λy.work-01(y) ∧ ARG0(A3(λz.person(z) ∧ name(z, A4(λw.name(w) ∧ op(w, AKKERMANS)))))∧ A1(ARG1(y, A5(λz.thing(z) ∧ name(z, A6(λw.name(w) ∧ op(w, STORIMANS)))))∧ op2(x, A7(λy.suffer-01 ∧ ARG0(y, R(3)) ∧ ARG1(y, A8(λw.injure-01(w) ∧ polarity(w, A9(λv.minor(v))))))))))

(a) Derivation of the final logical form. The intermediate categories and punctuation are omitted for brevity. To simplify the output, we mark conjunction with a single step. In practice it involves multiple binary steps.

```
(a/and
  :op1 (w/work
    :ARG0 (p/person
      :name (n/name :op1 "Akkermans")
    :ARG1 (t/thing
      :name (n2/name :op1 "Storimans")
  :op2 (i/suffer-01
    :ARG0 p
    :ARG1 (i/injury-01
      :polarity(m/minor)))
```

(b) Predicted AMR.

```
(a/and
  :op1 (w/work
    :ARG0 (p/person
      :name (n/name :op1 "Akkermans")
    :ARG3 (p2/person
      :name (n2/name :op1 "Storimans")
  :op2 (i/injure-01
    :ARG1 p
    :degree (m/minor)))
```

(c) Annotated AMR.

Figure 2: The max-scoring derivation for the sentence *Akkermans was working with Storimans and suffered minor injuries*. The system correctly analyzed the coordination, but it incorrectly introduced a suffer-01 predicate and inferred that *Storimans* fills the role of a job rather than a coworker. The SMATCH F1 for this example is 0.70.

The test caused protests from the United States and other nations.

Gold AMR	**JAMR** (SMATCH F1 : 0.72)	**Our System** (SMATCH F1 : 0.78)
(c/cause-01	(c/cause-01	(c/cause-01
:ARG0 (t/test)	:ARG0 (t/test-01	:ARG0 (t/test-01)
:ARG1 (p/protest-01	:ARG1 (o/other))	:ARG1 (a/and
:ARG0 (a/and	:ARG1 (p/protest-01	:op1 (p/protest-01
:op1 (c2/country	:ARG0 (a/and	:ARG0 (c2/country
:name (n/name	:op1 (c2/country	:name (n/name
:op1 “United”	:name (n/name	:op1 “United”
:op2 “States”))	:op1 “States”	:op2 “States”))
:op2 (n2/nation	:op2 “United”))	:op2 (n2/nation
:mod (o/other)))))	:op2 (n2/nation))))	:mod (o/other)))))

Figure 3: An example where JAMR recovered a more accurate global interpretation of the sentence, while our system scoped the coordination incorrectly (inferring that *caused* is an event with *the test* and *other nations* as arguments). However, our system shows stronger local coherence given the interpretation.

China has built and launched a communications satellite for Nigeria.

Gold AMR	**JAMR** (SMATCH F1 : 0.61)	**Our System** (SMATCH F1 : 0.88)
(a/and	(a/and	(a/and
:op1 (b/build-01	:op1 (h/have-03	:op1 (b/build-01
:ARG0 (c/country	:ARG0 (c/country	:ARG1 (c/country
:name (n/name	:name (n/name	:name (n/name
:op1 “China”))	:op1 “Nigeria”)))	:op1 “China”)))
:ARG1 (s/satellite	:op2 (b/build-01	:op2 (l/launch-01
:purpose (c2/communicate-01)	:ARG1 (s/satellite	:ARG1 (s/satellite
:beneficiary (c3/country	:ARG0-of (c2/communicate-01)	:poss (c2/communicate-01)
:name (n2/name	:ARG1-of (l/launch-01))))	:beneficiary (c3/country
:op1 “Nigeria”))))		:name (n2/name
:op2 (l/launch-01		:op1 “Nigeria”)))
:ARG0 c		:ARG0 c))
:ARG1 s))		

Figure 4: An example with a cyclic AMR. The country and the satellite are both co-referred by the build-01 and launch-01 events. Our system fails to recover the correct arguments for the build-01 event.

A key European arms control treaty must be maintained.

Gold AMR	**JAMR** (SMATCH F1 : 0.48)	**Our System** (SMATCH F1 : 0.94)
(o/obligate-01	(m/maintain-01	(o/obligate-01
:ARG2 (m/maintain-01	:ARG1 (t/treaty	:ARG2 (m/maintain-01
:ARG1 (t/treaty	:mod (k/key)	:ARG1 (t/treaty
:ARG0-of (c/control-01	:topic (c/control-01)))	:topic (c/control
:ARG1 (a/arm))		:ARG1 (a/arm))
:mod (k/key)		:mod (k/key)
:mod (c2/continent		:mod (c2/continent
:name (n/name		:name (n/name
:op1 “Europe”)))))		:op1 “Europe”)))))

Figure 5: An example where many of the core arguments of the events are not explicitly stated in the sentence. Our system is able to recover the overall structure, but interprets *arms control* as a topic rather than an event.

The principal cause is the use of drugs.

Gold AMR	**JAMR** (SMATCH F1 : 0.62)	**Our System** (SMATCH F1 : 0.43)
(c/cause-01	(c/cause-01	(u/use-01
:ARG0 (u/use-01	:ARG0-of (u/use-01	:ARG1 (d/drug)
:ARG1 (d/drug))	:ARG1 (d/drug	:ARG1 (c/cause))
:mod (p/principal))	:ARG2-of (i/include-91))))	

Figure 6: An example where the focus is on the subject rather than the event due to the copular. Due to the irregular compositional structure often assigned for such sentences in the AMR Bank, our system often fails to learn to recover the correct interpretation in such cases.

The body later was stolen from its crypt.

Gold AMR	**JAMR** (SMATCH F1 : 0.70)	**Our System** (SMATCH F1 : 0.67)
(s/steal-01	(s/steal-01	(s/steal-01
:ARG1 (b/body)	:ARG1 (b/body)	:ARG1 (t/thing
:ARG2 (c/crypt	:time (l/late))	:time (l/late
:poss b)		:degree (m/more))
:time (l/late		:mod (b/body)
:degree (m/more)))		:ARG2 t)

Figure 7: An example with a rare word, *crypt*, that cannot be handled by the named-entity recognizer. Our system performs word skipping in this example, causing it to recover a non-sensical AMR.

Tuxtepec is in the southeast side of Oaxaca.

Gold AMR	**JAMR** (SMATCH F1 : 0.27)	**Our System** (SMATCH F1 : 0.81)
(b/be-located-at-91	(s/side	(b/be-located-at-91
:ARG1 (c/city-district	:mod (s2/southeast)	:ARG1 (c/country
:name (n/name	:ARG1-of (i/include-91))	:name (n/name
:op1 "Tuxtepec"))		:op1 "Tuxtepec"))
:ARG2 (s/side		:ARG2 (s/side
:mod (s2/southeast)		:mod (s2/southeast)
:part-of (s3/state		:mod (c2/country
:name (n2/name		:name (n2/name
:op1 "Oaxaca")))))		:op1 "Oaxaca"))))

Figure 8: An example where world knowledge is needed to completely recover the AMR. Our system maintains a simple mapping from the named-entity recognizer types to AMR types and, by default, guesses that Tuxtepec and Oaxaca are countries, whereas they are respectively a city district and a state.

Iranian state television stated that Iranian police have killed 4 drug smugglers and have confiscated more than 1 ton of opium near the town of Mirjaveh.

Gold AMR	**JAMR** (SMATCH F1 : 0.61)	**Our System** (SMATCH F1 : 0.76)
(s/state-01	(s/state-01	(s/state-01
:ARG0 (t/television	:ARG0 (p/person	:ARG0 (t/television
:mod (s2/state)	:ARG0-of (h/have-org-role-91	:mod (s2/state)
:mod (c/country	:ARG1 (c/country	:mod (c/country
:name (n/name	:name (n/name	:name (n/name
:op1 "Iran")))	:op1 "Iran"))))	:op1 "Iran")))
:ARG1 (a/and	:ARG1 (a/and	:ARG1 (a/and
:op1 (k/kill-01	:op1 (k/kill-01	:op1 (k/kill-01
:ARG0 (p/police	:ARG0 (p2/police	:ARG0 (p/police)
:mod c)	:mod (c2/country	:ARG1 (p2/person
:ARG1 (p2/person	:name (n2/name	:quant 4
:ARG0-of (s3/smuggle-01	:op1 "Iran"))))	:ARG0-of (s3/smuggle-01
:quant 4	:ARG1 (p3/person	:ARG1 (d/drug))))
:ARG1 (d/drug)))	:ARG0-of (s2/smuggle-01	:op2 (c2/confiscate-01
:op2 (c3/confiscate-01	:ARG1 (d/drug)))	:ARG1 (p3/police)
:ARG0 p	:op2 (p4/person	:quant (m/more-than
:ARG1 (o/opium	:ARG0-of (h2/have-org-role-91))	:op1 (o/opium
:quant (m/more-than	:op3 (c3/confiscate-01	:quant (m2/mass-quantity
:quant (m2/mass-quantity	:ARG0 (t/television	:quant 1
:quant 1	:mod (s3/state))	:unit (t2/ton))))))
:unit (t2/ton)))))	:ARG1 (i/include-91	:location (n2/near
:location (n2/near	:ARG1 (o/opium	:op1 (t3/town
:location (t3/town	:quant (m/mass-quantity	:poss (p4/person))))
:name (n3/name	:unit (t2/ton)))	
:op1 "Mirjaveh"))))	:ARG3 (m2/more))	
	:ARG2 (t3/town)	
	:location (n3/near	
	:ARG1−of (i2/include-91)))))	

Figure 9: A longer example that is more typical of newswire text and demonstrates a variety of NLP problems, including word-sense disambiguation, co-reference resolution, named-entity recognition, coordination, quantifier scoping and prepositional phrase attachments. While our approach demonstrates progress, recovering a coherent complete structure for such longer and more complex sentences remains a challenge.