Chapter 13

Predicate-argument semantics

This chapter considers more "lightweight" semantic representations, which discard some aspects of first-order logic, but focus on predicate-argument structures. Let's begin by thinking about the semantics of events, with a simple example:

(13.1) Asha gives Boyang a book.

A first-order logical representation of this sentence is,

$$\exists x.\textsc{book}(x) \wedge \textsc{give}(\textsc{Asha}, \textsc{Boyang}, x) \qquad [13.1]$$

In this representation, we define variable x for the book, and we link the strings *Asha* and *Boyang* to entities ASHA and BOYANG. Because the action of giving involves a giver, a recipient, and a gift, the predicate GIVE must take three arguments.

Now suppose we have additional information about the event:

(13.2) Yesterday, Asha reluctantly gave Boyang a book.

One possible to solution is to extend the predicate GIVE to take additional arguments,

$$\exists x.\textsc{Book}(x) \wedge \textsc{Give}(\textsc{Asha}, \textsc{Boyang}, x, \textsc{yesterday}, \textsc{reluctantly}) \qquad [13.2]$$

But this is clearly unsatisfactory: *yesterday* and *relunctantly* are optional arguments, and we would need a different version of the GIVE predicate for every possible combination of arguments. **Event semantics** solves this problem by **reifying** the event as an existentially quantified variable e,

$$\exists e, x.\textsc{give-event}(e) \wedge \textsc{giver}(e, \textsc{Asha}) \wedge \textsc{gift}(e, x) \wedge \textsc{book}(e, x) \wedge \textsc{recipient}(e, \textsc{Boyang}) \\ \wedge \textsc{time}(e, \textsc{Yesterday}) \wedge \textsc{manner}(e, \textsc{reluctantly})$$

In this way, each argument of the event — the giver, the recipient, the gift — can be represented with a relation of its own, linking the argument to the event e. The expression $\textsc{giver}(e, \textsc{Asha})$ says that $\textsc{Asha}$ plays the **role** of $\textsc{giver}$ in the event. This reformulation handles the problem of optional information such as the time or manner of the event, which are called **adjuncts**. Unlike arguments, adjuncts are not a mandatory part of the relation, but under this representation, they can be expressed with additional logical relations that are conjoined to the semantic interpretation of the sentence. [1]

The event semantic representation can be applied to nested clauses, e.g.,

(13.3) Chris sees Asha pay Boyang.

This is done by using the event variable as an argument:

$$\begin{aligned}&\exists e_1 \exists e_2\, \textsc{See-Event}(e_1) \wedge \textsc{seer}(e_1, \textsc{Chris}) \wedge \textsc{sight}(e_1, e_2)\\ &\quad \wedge \textsc{Pay-Event}(e_2) \wedge \textsc{payer}(e_2, \textsc{Asha}) \wedge \textsc{payee}(e_2, \textsc{Boyang})\end{aligned} \qquad [13.3]$$

As with first-order logic, the goal of event semantics is to provide a representation that generalizes over many surface forms. Consider the following paraphrases of (13.1):

(13.4)
a. Asha gives a book to Boyang.
b. A book is given to Boyang by Asha.
c. A book is given by Asha to Boyang.
d. The gift of a book from Asha to Boyang ...

All have the same event semantic meaning as Equation 13.1, but the ways in which the meaning can be expressed are diverse. The final example does not even include a verb: events are often introduced by verbs, but as shown by (13.4d), the noun *gift* can introduce the same predicate, with the same accompanying arguments.

Semantic role labeling (SRL) is a relaxed form of semantic parsing, in which each semantic role is filled by a set of tokens from the text itself. This is sometimes called "shallow semantics" because, unlike model-theoretic semantic parsing, role fillers need not be symbolic expressions with denotations in some world model. A semantic role labeling system is required to identify all predicates, and then specify the spans of text that fill each role. To give a sense of the task, here is a more complicated example:

(13.5) Boyang wants Asha to give him a linguistics book.

[1] This representation is often called **Neo-Davidsonian event semantics**. The use of existentially-quantified event variables was proposed by Davidson (1967) to handle the issue of optional adjuncts. In Neo-Davidsonian semantics, this treatment of adjuncts is extended to mandatory arguments as well (e.g., Parsons, 1990).

In this example, there are two predicates, expressed by the verbs *want* and *give*. Thus, a semantic role labeler might return the following output:

- (PREDICATE : *wants*, WANTER : *Boyang*, DESIRE : *Asha to give him a linguistics book*)
- (PREDICATE : *give*, GIVER : *Asha*, RECIPIENT : *him*, GIFT : *a linguistics book*)

Boyang and *him* may refer to the same person, but the semantic role labeling is not required to resolve this reference. Other predicate-argument representations, such as **Abstract Meaning Representation (AMR)**, do require reference resolution. We will return to AMR in § 13.3, but first, let us further consider the definition of semantic roles.

13.1 Semantic roles

In event semantics, it is necessary to specify a number of additional logical relations to link arguments to events: GIVER, RECIPIENT, SEER, SIGHT, etc. Indeed, every predicate requires a set of logical relations to express its own arguments. In contrast, adjuncts such as TIME and MANNER are shared across many types of events. A natural question is whether it is possible to treat mandatory arguments more like adjuncts, by identifying a set of generic argument types that are shared across many event predicates. This can be further motivated by examples involving related verbs:

(13.6) a. Asha gave Boyang a book.
b. Asha loaned Boyang a book.
c. Asha taught Boyang a lesson.
d. Asha gave Boyang a lesson.

The respective roles of Asha, Boyang, and the book are nearly identical across the first two examples. The third example is slightly different, but the fourth example shows that the roles of GIVER and TEACHER can be viewed as related.

One way to think about the relationship between roles such as GIVER and TEACHER is by enumerating the set of properties that an entity typically possesses when it fulfills these roles: givers and teachers are usually **animate** (they are alive and sentient) and **volitional** (they choose to enter into the action).[2] In contrast, the thing that gets loaned or taught is usually not animate or volitional; furthermore, it is unchanged by the event.

Building on these ideas, **thematic roles** generalize across predicates by leveraging the shared semantic properties of typical role fillers (Fillmore, 1968). For example, in examples (13.6a-13.6d), Asha plays a similar role in all four sentences, which we will call the

[2]There are always exceptions. For example, in the sentence *The C programming language has taught me a lot about perseverance*, the "teacher" is the *The C programming language*, which is presumably not animate or volitional.

	Asha	*gave*	*Boyang*	*a book*
VerbNet	AGENT		RECIPIENT	THEME
PropBank	ARG0: giver		ARG2: entity given to	ARG1: thing given
FrameNet	DONOR		RECIPIENT	THEME
	Asha	*taught*	*Boyang*	*algebra*
VerbNet	AGENT		RECIPIENT	TOPIC
PropBank	ARG0: teacher		ARG2: student	ARG1: subject
FrameNet	TEACHER		STUDENT	SUBJECT

Figure 13.1: Example semantic annotations according to VerbNet, PropBank, and FrameNet

agent. This reflects several shared semantic properties: she is the one who is actively and intentionally performing the action, while Boyang is a more passive participant; the book and the lesson would play a different role, as non-animate participants in the event.

Example annotations from three well known systems are shown in Figure 13.1. We will now discuss these systems in more detail.

13.1.1 VerbNet

VerbNet (Kipper-Schuler, 2005) is a lexicon of verbs, and it includes thirty "core" thematic roles played by arguments to these verbs. Here are some example roles, accompanied by their definitions from the VerbNet Guidelines.[3]

- AGENT: "ACTOR in an event who initiates and carries out the event intentionally or consciously, and who exists independently of the event."
- PATIENT: "UNDERGOER in an event that experiences a change of state, location or condition, that is causally involved or directly affected by other participants, and exists independently of the event."
- RECIPIENT: "DESTINATION that is animate"
- THEME: "UNDERGOER that is central to an event or state that does not have control over the way the event occurs, is not structurally changed by the event, and/or is characterized as being in a certain position or condition throughout the state."
- TOPIC: "THEME characterized by information content transferred to another participant."

[3]http://verbs.colorado.edu/verb-index/VerbNet_Guidelines.pdf

VerbNet roles are organized in a hierarchy, so that a TOPIC is a type of THEME, which in turn is a type of UNDERGOER, which is a type of PARTICIPANT, the top-level category.

In addition, VerbNet organizes verb senses into a class hierarchy, in which verb senses that have similar meanings are grouped together. Recall from § 4.2 that multiple meanings of the same word are called **senses**, and that WordNet identifies senses for many English words. VerbNet builds on WordNet, so that verb classes are identified by the WordNet senses of the verbs that they contain. For example, the verb class `give-13.1` includes the first WordNet sense of *loan* and the second WordNet sense of *lend*.

Each VerbNet class or subclass takes a set of thematic roles. For example, `give-13.1` takes arguments with the thematic roles of AGENT, THEME, and RECIPIENT;[4] the predicate TEACH takes arguments with the thematic roles AGENT, TOPIC, RECIPIENT, and SOURCE.[5] So according to VerbNet, *Asha* and *Boyang* play the roles of AGENT and RECIPIENT in the sentences,

(13.7) a. Asha gave Boyang a book.
b. Asha taught Boyang algebra.

The *book* and *algebra* are both THEMES, but *algebra* is a subcategory of THEME — a TOPIC — because it consists of information content that is given to the receiver.

13.1.2 Proto-roles and PropBank

Detailed thematic role inventories of the sort used in VerbNet are not universally accepted. For example, Dowty (1991, pp. 547) notes that "Linguists have often found it hard to agree on, and to motivate, the location of the boundary between role types." He argues that a solid distinction can be identified between just two **proto-roles**:

Proto-Agent. Characterized by volitional involvement in the event or state; sentience and/or perception; causing an event or change of state in another participant; movement; exists independently of the event.

Proto-Patient. Undergoes change of state; causally affected by another participant; stationary relative to the movement of another participant; does not exist independently of the event.[6]

[4] `https://verbs.colorado.edu/verb-index/vn/give-13.1.php`

[5] `https://verbs.colorado.edu/verb-index/vn/transfer_mesg-37.1.1.php`

[6] Reisinger et al. (2015) ask crowd workers to annotate these properties directly, finding that annotators tend to agree on the properties of each argument. They also find that in English, arguments having more proto-agent properties tend to appear in subject position, while arguments with more proto-patient properties appear in object position.

In the examples in Figure 13.1, Asha has most of the proto-agent properties: in giving the book to Boyang, she is acting volitionally (as opposed to *Boyang got a book from Asha*, in which it is not clear whether Asha gave up the book willingly); she is sentient; she causes a change of state in Boyang; she exists independently of the event. Boyang has some proto-agent properties: he is sentient and exists independently of the event. But he also has some proto-patient properties: he is the one who is causally affected and who undergoes change of state. The book that Asha gives Boyang has even fewer of the proto-agent properties: it is not volitional or sentient, and it has no causal role. But it also lacks many of the proto-patient properties: it does not undergo change of state, exists independently of the event, and is not stationary.

The **Proposition Bank**, or PropBank (Palmer et al., 2005), builds on this basic agent-patient distinction, as a middle ground between generic thematic roles and roles that are specific to each predicate. Each verb is linked to a list of numbered arguments, with ARG0 as the proto-agent and ARG1 as the proto-patient. Additional numbered arguments are verb-specific. For example, for the predicate TEACH,[7] the arguments are:

- ARG0: the teacher
- ARG1: the subject
- ARG2: the student(s)

Verbs may have any number of arguments: for example, WANT and GET have five, while EAT has only ARG0 and ARG1. In addition to the semantic arguments found in the frame files, roughly a dozen general-purpose adjuncts may be used in combination with any verb. These are shown in Table 13.1.

PropBank-style semantic role labeling is annotated over the entire Penn Treebank. This annotation includes the sense of each verbal predicate, as well as the argument spans.

13.1.3 FrameNet

Semantic **frames** are descriptions of situations or events. Frames may be *evoked* by one of their **lexical units** (often a verb, but not always), and they include some number of **frame elements**, which are like roles (Fillmore, 1976). For example, the act of teaching is a frame, and can be evoked by the verb *taught*; the associated frame elements include the teacher, the student(s), and the subject being taught. Frame semantics has played a significant role in the history of artificial intelligence, in the work of Minsky (1974) and Schank and Abelson (1977). In natural language processing, the theory of frame semantics has been implemented in **FrameNet** (Fillmore and Baker, 2009), which consists of a lexicon

[7]http://verbs.colorado.edu/propbank/framesets-english-aliases/teach.html

TMP	time	*Boyang ate a bagel* [$_{\text{AM-TMP}}$ *yesterday*].
LOC	location	*Asha studies in* [$_{\text{AM-LOC}}$ *Stuttgart*]
MOD	modal verb	*Asha* [$_{\text{AM-MOD}}$ *will*] *study in Stuttgart*
ADV	general purpose	[$_{\text{AM-ADV}}$ *Luckily*], *Asha knew algebra.*
MNR	manner	*Asha ate* [$_{\text{AM-MNR}}$ *aggressively*].
DIS	discourse connective	[$_{\text{AM-DIS}}$ *However*], *Asha prefers algebra.*
PRP	purpose	*Barry studied* [$_{\text{AM-PRP}}$ *to pass the bar*].
DIR	direction	*Workers dumped burlap sacks* [$_{\text{AM-DIR}}$ *into a bin*].
NEG	negation	*Asha does* [$_{\text{AM-NEG}}$ *not*] *speak Albanian.*
EXT	extent	*Prices increased* [$_{\text{AM-EXT}}$ *4%*].
CAU	cause	*Boyang returned the book* [$_{\text{AM-CAU}}$ *because it was overdue*].

Table 13.1: PropBank adjuncts (Palmer et al., 2005), sorted by frequency in the corpus

of roughly 1000 frames, and a corpus of more than 200,000 "exemplar sentences," in which the frames and their elements are annotated.[8]

Rather than seeking to link semantic roles such as TEACHER and GIVER into thematic roles such as AGENT, FrameNet aggressively groups verbs into frames, and links semantically-related roles across frames. For example, the following two sentences would be annotated identically in FrameNet:

(13.8) a. Asha taught Boyang algebra.

b. Boyang learned algebra from Asha.

This is because *teach* and *learn* are both lexical units in the EDUCATION_TEACHING frame. Furthermore, roles can be shared even when the frames are distinct, as in the following two examples:

(13.9) a. Asha gave Boyang a book.

b. Boyang got a book from Asha.

The GIVING and GETTING frames both have RECIPIENT and THEME elements, so Boyang and the book would play the same role. Asha's role is different: she is the DONOR in the GIVING frame, and the SOURCE in the GETTING frame. FrameNet makes extensive use of multiple inheritance to share information across frames and frame elements: for example, the COMMERCE_SELL and LENDING frames inherit from GIVING frame.

[8]Current details and data can be found at `https://framenet.icsi.berkeley.edu/`

13.2 Semantic role labeling

The task of semantic role labeling is to identify the parts of the sentence comprising the semantic roles. In English, this task is typically performed on the PropBank corpus, with the goal of producing outputs in the following form:

(13.10) $[_{\text{ARG0}}$ Asha$]$ $[_{\text{GIVE.01}}$ gave$]$ $[_{\text{ARG2}}$ Boyang's mom$]$ $[_{\text{ARG1}}$ a book$]$ $[_{\text{AM-TMP}}$ yesterday$]$.

Note that a single sentence may have multiple verbs, and therefore a given word may be part of multiple role-fillers:

(13.11) $[_{\text{ARG0}}$ Asha$]$ $[_{\text{WANT.01}}$ wanted$]$
Asha wanted
$[_{\text{ARG1}}$ Boyang to give her the book$]$.
$[_{\text{ARG0}}$ Boyang$]$ $[_{\text{GIVE.01}}$ to give$]$ $[_{\text{ARG2}}$ her$]$ $[_{\text{ARG1}}$ the book$]$.

13.2.1 Semantic role labeling as classification

PropBank is annotated on the Penn Treebank, and annotators used phrasal constituents (§ 9.2.2) to fill the roles. PropBank semantic role labeling can be viewed as the task of assigning to each phrase a label from the set $\mathcal{R} = \{\varnothing, \text{PRED}, \text{ARG0}, \text{ARG1}, \text{ARG2}, \ldots, \text{AM-LOC}, \text{AM-TMP}, .$
with respect to each predicate. If we treat semantic role labeling as a classification problem, we obtain the following functional form:

$$\hat{y}_{(i,j)} = \underset{y}{\operatorname{argmax}}\, \psi(\boldsymbol{w}, y, i, j, \rho, \tau), \qquad [13.4]$$

where,

- (i, j) indicates the span of a phrasal constituent $(w_{i+1}, w_{i+2}, \ldots, w_j)$;[9]
- $\boldsymbol{w}$ represents the sentence as a sequence of tokens;
- ρ is the index of the predicate verb in $\boldsymbol{w}$;
- τ is the structure of the phrasal constituent parse of $\boldsymbol{w}$.

Early work on semantic role labeling focused on discriminative feature-based models, where $\psi(\boldsymbol{w}, y, i, j, \rho, \tau) = \boldsymbol{\theta} \cdot \boldsymbol{f}(\boldsymbol{w}, y, i, j, \rho, \tau)$. Table 13.2 shows the features used in a seminal paper on FrameNet semantic role labeling (Gildea and Jurafsky, 2002). By 2005 there

[9]PropBank roles can also be filled by **split constituents**, which are discontinuous spans of text. This situation most frequently in reported speech, e.g. $[_{\text{ARG1}}$ *By addressing these problems*$]$, *Mr. Maxwell said,* $[_{\text{ARG1}}$ *the new funds have become extremely attractive.*$]$ (example adapted from Palmer et al., 2005). This issue is typically addressed by defining "continuation arguments", e.g. C-ARG1, which refers to the continuation of ARG1 after the split.

Predicate lemma and POS tag	The lemma of the predicate verb and its part-of-speech tag
Voice	Whether the predicate is in active or passive voice, as determined by a set of syntactic patterns for identifying passive voice constructions
Phrase type	The constituent phrase type for the proposed argument in the parse tree, e.g. NP, PP
Headword and POS tag	The head word of the proposed argument and its POS tag, identified using the Collins (1997) rules
Position	Whether the proposed argument comes before or after the predicate in the sentence
Syntactic path	The set of steps on the parse tree from the proposed argument to the predicate (described in detail in the text)
Subcategorization	The syntactic production from the first branching node above the predicate. For example, in Figure 13.2, the subcategorization feature around *taught* would be VP $\rightarrow$ VBD NP PP.

Table 13.2: Features used in semantic role labeling by Gildea and Jurafsky (2002).

were several systems for PropBank semantic role labeling, and their approaches and feature sets are summarized by Carreras and Màrquez (2005). Typical features include: the phrase type, head word, part-of-speech, boundaries, and neighbors of the proposed argument $\boldsymbol{w}_{i+1:j}$; the word, lemma, part-of-speech, and voice of the verb w_ρ (active or passive), as well as features relating to its frameset; the distance and path between the verb and the proposed argument. In this way, semantic role labeling systems are high-level "consumers" in the NLP stack, using features produced from lower-level components such as part-of-speech taggers and parsers. More comprehensive feature sets are enumerated by Das et al. (2014) and Täckström et al. (2015).

A particularly powerful class of features relate to the **syntactic path** between the argument and the predicate. These features capture the sequence of moves required to get from the argument to the verb by traversing the phrasal constituent parse of the sentence. The idea of these features is to capture syntactic regularities in how various arguments are realized. Syntactic path features are best illustrated by example, using the parse tree in Figure 13.2:

- The path from *Asha* to the verb *taught* is NNP↑NP↑S↓VP↓VBD. The first part of the path, NNP↑NP↑S, means that we must travel up the parse tree from the NNP tag (proper noun) to the S (sentence) constituent. The second part of the path, S↓VP↓VBD, means that we reach the verb by producing a VP (verb phrase) from

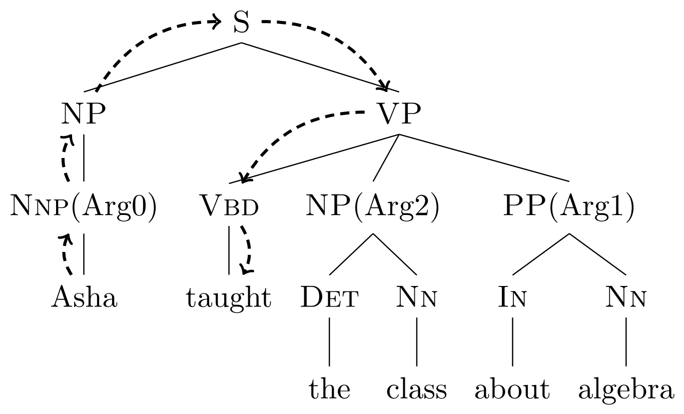

Figure 13.2: Semantic role labeling on the phrase-structure parse tree for a sentence. The dashed line indicates the syntactic path from *Asha* to the predicate verb *taught*.

the S constituent, and then by producing a VBD (past tense verb). This feature is consistent with *Asha* being in subject position, since the path includes the sentence root S.

- The path from *the class* to *taught* is NP↑VP↓VBD. This is consistent with *the class* being in object position, since the path passes through the VP node that dominates the verb *taught*.

Because there are many possible path features, it can also be helpful to look at smaller parts: for example, the upward and downward parts can be treated as separate features; another feature might consider whether S appears anywhere in the path.

Rather than using the constituent parse, it is also possible to build features from the **dependency path** (see § 11.4) between the head word of each argument and the verb (Pradhan et al., 2005). Using the Universal Dependency part-of-speech tagset and dependency relations (Nivre et al., 2016), the dependency path from *Asha* to *taught* is PROPN $\underset{\text{NSUBJ}}{\leftarrow}$ VERB, because *taught* is the head of a relation of type $\underset{\text{NSUBJ}}{\leftarrow}$ with *Asha*. Similarly, the dependency path from *class* to *taught* is NOUN $\underset{\text{DOBJ}}{\leftarrow}$ VERB, because *class* heads the noun phrase that is a direct object of *taught*. A more interesting example is *Asha wanted to teach the class*, where the path from *Asha* to *teach* is PROPN $\underset{\text{NSUBJ}}{\leftarrow}$ VERB $\underset{\text{XCOMP}}{\rightarrow}$ VERB. The right-facing arrow in second relation indicates that *wanted* is the head of its XCOMP relation with *teach*.

13.2.2 Semantic role labeling as constrained optimization

A potential problem with treating SRL as a classification problem is that there are a number of sentence-level **constraints**, which a classifier might violate.

- For a given verb, there can be only one argument of each type (ARG0, ARG1, etc.)
- Arguments cannot overlap. This problem arises when we are labeling the phrases in a constituent parse tree, as shown in Figure 13.2: if we label the PP *about algebra* as an argument or adjunct, then its children *about* and *algebra* must be labeled as $\varnothing$. The same constraint also applies to the syntactic ancestors of this phrase.

These constraints introduce dependencies across labeling decisions. In structure prediction problems such as sequence labeling and parsing, such dependencies are usually handled by defining a scoring over the entire structure, $\boldsymbol{y}$. Efficient inference requires that the global score decomposes into local parts: for example, in sequence labeling, the scoring function decomposes into scores of pairs of adjacent tags, permitting the application of the Viterbi algorithm for inference. But the constraints that arise in semantic role labeling are less amenable to local decomposition.[10] We therefore consider **constrained optimization** as an alternative solution.

Let the set $\mathcal{C}(\tau)$ refer to all labelings that obey the constraints introduced by the parse τ. The semantic role labeling problem can be reformulated as a constrained optimization over $\boldsymbol{y} \in \mathcal{C}(\tau)$,

$$\begin{aligned} \max_{\boldsymbol{y}} \quad & \sum_{(i,j)\in\tau} \psi(\boldsymbol{w}, y_{i,j}, i, j, \rho, \tau) \\ \text{s.t.} \quad & \boldsymbol{y} \in \mathcal{C}(\tau). \end{aligned} \tag{13.5}$$

In this formulation, the objective (shown on the first line) is a separable function of each individual labeling decision, but the constraints (shown on the second line) apply to the overall labeling. The sum $\sum_{(i,j)\in\tau}$ indicates that we are summing over all constituent spans in the parse τ. The expression s.t. in the second line means that we maximize the objective *subject to* the constraint $\boldsymbol{y} \in \mathcal{C}(\tau)$.

A number of practical algorithms exist for restricted forms of constrained optimization. One such restricted form is **integer linear programming**, in which the objective and constraints are linear functions of integer variables. To formulate SRL as an integer linear program, we begin by rewriting the labels as a set of binary variables $\boldsymbol{z} = \{z_{i,j,r}\}$ (Punyakanok et al., 2008),

$$z_{i,j,r} = \begin{cases} 1, & y_{i,j} = r \\ 0, & \text{otherwise,} \end{cases} \tag{13.6}$$

[10]Dynamic programming solutions have been proposed by Tromble and Eisner (2006) and Täckström et al. (2015), but they involves creating a trellis structure whose size is exponential in the number of labels.

where $r \in \mathcal{R}$ is a label in the set $\{\text{ARG0}, \text{ARG1}, \ldots, \text{AM-LOC}, \ldots, \varnothing\}$. Thus, the variables $\boldsymbol{z}$ are a binarized version of the semantic role labeling $\boldsymbol{y}$.

The objective can then be formulated as a linear function of $\boldsymbol{z}$.

$$\sum_{(i,j)\in\tau} \psi(\boldsymbol{w}, y_{i,j}, i, j, \rho, \tau) = \sum_{i,j,r} \psi(\boldsymbol{w}, r, i, j, \rho, \tau) \times z_{i,j,r}, \quad [13.7]$$

which is the sum of the scores of all relations, as indicated by $z_{i,j,r}$.

Constraints Integer linear programming permits linear inequality constraints, of the general form $\mathbf{A}\boldsymbol{z} \leq \boldsymbol{b}$, where the parameters $\mathbf{A}$ and $\boldsymbol{b}$ define the constraints. To make this more concrete, let's start with the constraint that each non-null role type can occur only once in a sentence. This constraint can be written,

$$\forall r \neq \varnothing, \quad \sum_{(i,j)\in\tau} z_{i,j,r} \leq 1. \quad [13.8]$$

Recall that $z_{i,j,r} = 1$ iff the span (i, j) has label r; this constraint says that for each possible label $r \neq \varnothing$, there can be at most one (i, j) such that $z_{i,j,r} = 1$. Rewriting this constraint can be written in the form $\mathbf{A}\boldsymbol{z} \leq \boldsymbol{b}$, as you will find if you complete the exercises at the end of the chapter.

Now consider the constraint that labels cannot overlap. Let's define the convenience function $o((i, j), (i', j')) = 1$ iff (i, j) overlaps (i', j'), and zero otherwise. Thus, o will indicate if a constituent (i', j') is either an ancestor or descendant of (i, j). The constraint is that if two constituents overlap, only one can have a non-null label:

$$\forall (i, j) \in \tau, \quad \sum_{(i',j')\in\tau} \sum_{r\neq\varnothing} o((i, j), (i', j')) \times z_{i',j',r} \leq 1, \quad [13.9]$$

where $o((i, j), (i, j)) = 1$.

In summary, the semantic role labeling problem can thus be rewritten as the following integer linear program,

$$\max_{\boldsymbol{z}\in\{0,1\}^{|\tau|}} \quad \sum_{(i,j)\in\tau} \sum_{r\in\mathcal{R}} z_{i,j,r}\psi_{i,j,r} \quad [13.10]$$

$$s.t. \quad \forall r \neq \varnothing, \quad \sum_{(i,j)\in\tau} z_{i,j,r} \leq 1. \quad [13.11]$$

$$\forall (i, j) \in \tau, \quad \sum_{(i',j')\in\tau} \sum_{r\neq\varnothing} o((i, j), (i', j')) \times z_{i',j',r} \leq 1. \quad [13.12]$$

Learning with constraints Learning can be performed in the context of constrained optimization using the usual perceptron or large-margin classification updates. Because constrained inference is generally more time-consuming, a key question is whether it is necessary to apply the constraints during learning. Chang et al. (2008) find that better performance can be obtained by learning *without* constraints, and then applying constraints only when using the trained model to predict semantic roles for unseen data.

How important are the constraints? Das et al. (2014) find that an unconstrained, classification-based method performs nearly as well as constrained optimization for FrameNet parsing: while it commits many violations of the "no-overlap" constraint, the overall F_1 score is less than one point worse than the score at the constrained optimum. Similar results were obtained for PropBank semantic role labeling by Punyakanok et al. (2008). He et al. (2017) find that constrained inference makes a bigger impact if the constraints are based on manually-labeled "gold" syntactic parses. This implies that errors from the syntactic parser may limit the effectiveness of the constraints. Punyakanok et al. (2008) hedge against parser error by including constituents from several different parsers; any constituent can be selected from any parse, and additional constraints ensure that overlapping constituents are not selected.

Implementation Integer linear programming solvers such as `glpk`,[11] `cplex`,[12] and `Gurobi`[13] allow inequality constraints to be expressed directly in the problem definition, rather than in the matrix form $\mathbf{A}\boldsymbol{z} \leq \boldsymbol{b}$. The time complexity of integer linear programming is theoretically exponential in the number of variables $|\boldsymbol{z}|$, but in practice these off-the-shelf solvers obtain good solutions efficiently. Using a standard desktop computer, Das et al. (2014) report that the `cplex` solver requires 43 seconds to perform inference on the FrameNet test set, which contains 4,458 predicates.

Recent work has shown that many constrained optimization problems in natural language processing can be solved in a highly parallelized fashion, using optimization techniques such as **dual decomposition**, which are capable of exploiting the underlying problem structure (Rush et al., 2010). Das et al. (2014) apply this technique to FrameNet semantic role labeling, obtaining an order-of-magnitude speedup over `cplex`.

13.2.3 Neural semantic role labeling

Neural network approaches to SRL have tended to treat it as a sequence labeling task, using a labeling scheme such as the **BIO notation**, which we previously saw in named entity recognition (§ 8.3). In this notation, the first token in a span of type ARG1 is labeled

[11] https://www.gnu.org/software/glpk/
[12] https://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/
[13] http://www.gurobi.com/

B-ARG1; all remaining tokens in the span are *inside*, and are therefore labeled I-ARG1. Tokens outside any argument are labeled O. For example:

(13.12)	*Asha*	*taught*	*Boyang*	*'s*	*mom*	*about*	*algebra*
	B-ARG0	PRED	B-ARG2	I-ARG2	I-ARG2	B-ARG1	I-ARG1

Recurrent neural networks (§ 7.6) are a natural approach to this tagging task. For example, Zhou and Xu (2015) apply a deep bidirectional multilayer LSTM (see § 7.6) to PropBank semantic role labeling. In this model, each bidirectional LSTM serves as input for another, higher-level bidirectional LSTM, allowing complex non-linear transformations of the original input embeddings, $\mathbf{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_M]$. The hidden state of the final LSTM is $\mathbf{Z}^{(K)} = [\boldsymbol{z}_1^{(K)}, \boldsymbol{z}_2^{(K)}, \ldots, \boldsymbol{z}_M^{(K)}]$. The "emission" score for each tag $Y_m = y$ is equal to the inner product $\boldsymbol{\theta}_y \cdot \boldsymbol{z}_m^{(K)}$, and there is also a transition score for each pair of adjacent tags. The complete model can be written,

$$\mathbf{Z}^{(1)} = \text{BiLSTM}(\mathbf{X}) \tag{13.13}$$

$$\mathbf{Z}^{(i)} = \text{BiLSTM}(\mathbf{Z}^{(i-1)}) \tag{13.14}$$

$$\hat{\boldsymbol{y}} = \underset{\boldsymbol{y}}{\text{argmax}} \sum_{m-1}^{M} \boldsymbol{\Theta}^{(y)} \boldsymbol{z}_m^{(K)} + \psi_{y_{m-1}, y_m}. \tag{13.15}$$

Note that the final step maximizes over the entire labeling $\boldsymbol{y}$, and includes a score for each tag transition ψ_{y_{m-1}, y_m}. This combination of LSTM and pairwise potentials on tags is an example of an **LSTM-CRF**. The maximization over $\boldsymbol{y}$ is performed by the Viterbi algorithm.

This model strongly outperformed alternative approaches at the time, including constrained decoding and convolutional neural networks.[14] More recent work has combined recurrent neural network models with constrained decoding, using the A^* search algorithm to search over labelings that are feasible with respect to the constraints (He et al., 2017). This yields small improvements over the method of Zhou and Xu (2015). He et al. (2017) obtain larger improvements by creating an **ensemble** of SRL systems, each trained on an 80% subsample of the corpus. The average prediction across this ensemble is more robust than any individual model.

13.3 Abstract Meaning Representation

Semantic role labeling transforms the task of semantic parsing to a labeling task. Consider the sentence,

[14] The successful application of **convolutional neural networks** to semantic role labeling by Collobert and Weston (2008) was an influential early result in the current wave of neural networks in natural language processing.

```
(w / want-01
  :ARG0 (h / whale)
  :ARG1 (p / pursue-02
          :ARG0 (c / captain)
          :ARG1 h))
```

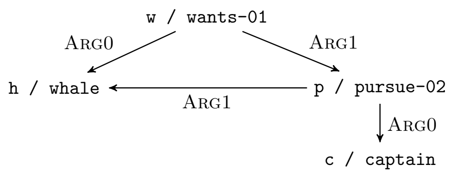

Figure 13.3: Two views of the AMR representation for the sentence *The whale wants the captain to pursue him.*

(13.13) The whale wants the captain to pursue him.

The PropBank semantic role labeling analysis is:

- (PREDICATE : *wants*, ARG0 : *the whale*, ARG1 : *the captain to pursue him*)
- (PREDICATE : *pursue*, ARG0 : *the captain*, ARG1 : *him*)

The **Abstract Meaning Representation (AMR)** unifies this analysis into a graph structure, in which each node is a **variable**, and each edge indicates a **concept** (Banarescu et al., 2013). This can be written in two ways, as shown in Figure 13.3. On the left is the PENMAN notation (Matthiessen and Bateman, 1991), in which each set of parentheses introduces a variable. Each variable is an **instance** of a concept, which is indicated with the slash notation: for example, `w / want-01` indicates that the variable `w` is an instance of the concept `want-01`, which in turn refers to the PropBank frame for the first sense of the verb *want*; `pursue-02` refers to the second sense of *pursue*. Relations are introduced with colons: for example, `:ARG0 (c / captain)` indicates a relation of type ARG0 with the newly-introduced variable `c`. Variables can be reused, so that when the variable `h` appears again as an argument to `p`, it is understood to refer to the same whale in both cases. This arrangement is indicated compactly in the graph structure on the right, with edges indicating concepts.

One way in which AMR differs from PropBank-style semantic role labeling is that it reifies each entity as a variable: for example, the *whale* in (13.13) is reified in the variable `h`, which is reused as ARG0 in its relationship with `w / want-01`, and as ARG1 in its relationship with `p / pursue-02`. Reifying entities as variables also makes it possible to represent the substructure of noun phrases more explicitly. For example, *Asha borrowed the algebra book* would be represented as:

```
(b / borrow-01
   :ARG0 (p / person
           :name (n / name
                   :op1 "Asha"))
```

```
    :ARG1 (b2 / book
             :topic (a / algebra)))
```

This indicates that the variable `p` is a person, whose `name` is the variable `n`; that name has one token, the string *Asha*. Similarly, the variable `b2` is a book, and the `topic` of `b2` is a variable `a` whose type is `algebra`. The relations `name` and `topic` are examples of "non-core roles", which are similar to adjunct modifiers in PropBank. However, AMR's inventory is more extensive, including more than 70 non-core roles, such as negation, time, manner, frequency, and location. Lists and sequences — such as the list of tokens in a name — are described using the roles `op1`, `op2`, etc.

Another feature of AMR is that a semantic predicate can be introduced by any syntactic element, as in the following examples from Banarescu et al. (2013):

(13.14) a. The boy destroyed the room.
b. the destruction of the room by the boy ...
c. the boy's destruction of the room ...

All these examples have the same semantics in AMR,

```
(d / destroy-01
   :ARG0 (b / boy)
   :ARG1 (r / room))
```

The noun *destruction* is linked to the verb *destroy*, which is captured by the PropBank frame `destroy-01`. This can happen with adjectives as well: in the phrase *the attractive spy*, the adjective *attractive* is linked to the PropBank frame `attract-01`:

```
(s / spy
   :ARG0-of (a / attract-01))
```

In this example, `ARG0-of` is an **inverse relation**, indicating that `s` is the `ARG0` of the predicate `a`. Inverse relations make it possible for all AMR parses to have a single root concept.

While AMR goes farther than semantic role labeling, it does not link semantically-related frames such as `buy/sell` (as FrameNet does). AMR also does not handle quantification (as first-order predicate calculus does), and it makes no attempt to handle noun number and verb tense (as PropBank does).

13.3.1 AMR Parsing

Abstract Meaning Representation is not a labeling of the original text — unlike PropBank semantic role labeling, and most of the other tagging and parsing tasks that we have encountered thus far. The AMR for a given sentence may include multiple concepts for single words in the sentence: as we have seen, the sentence *Asha likes algebra* contains both `person` and `name` concepts for the word *Asha*. Conversely, words in the sentence may not appear in the AMR: in *Boyang made a tour of campus*, the **light verb** *make* would not appear in the AMR, which would instead be rooted on the predicate `tour`. As a result, AMR is difficult to parse, and even evaluating AMR parsing involves considerable algorithmic complexity (Cai and Yates, 2013).

A further complexity is that AMR labeled datasets do not explicitly show the **alignment** between the AMR annotation and the words in the sentence. For example, the link between the word *wants* and the concept `want-01` is not annotated. To acquire training data for learning-based parsers, it is therefore necessary to first perform an alignment between the training sentences and their AMR parses. Flanigan et al. (2014) introduce a rule-based parser, which links text to concepts through a series of increasingly high-recall steps.

As with dependency parsing, AMR can be parsed by graph-based methods that explore the space of graph structures, or by incremental transition-based algorithms. One approach to graph-based AMR parsing is to first group adjacent tokens into local substructures, and then to search the space of graphs over these substructures (Flanigan et al., 2014). The identification of concept subgraphs can be formulated as a sequence labeling problem, and the subsequent graph search can be solved using integer linear programming (§ 13.2.2). Various transition-based parsing algorithms have been proposed. Wang et al. (2015) construct an AMR graph by incrementally modifying the *syntactic* dependency graph. At each step, the parser performs an action: for example, adding an AMR relation label to the current dependency edge, swapping the direction of a syntactic dependency edge, or cutting an edge and reattaching the orphaned subtree to a new parent.

Additional resources

Practical semantic role labeling was first made possible by the PropBank annotations on the Penn Treebank (Palmer et al., 2005). Abend and Rappoport (2017) survey several semantic representation schemes, including semantic role labeling and AMR. Other linguistic features of AMR are summarized in the original paper (Banarescu et al., 2013) and the tutorial slides by Schneider et al. (2015). Recent shared tasks have undertaken semantic dependency parsing, in which the goal is to identify semantic relationships between pairs of words (Oepen et al., 2014); see Ivanova et al. (2012) for an overview of connections between syntactic and semantic dependencies.

Exercises

1. Write out an event semantic representation for the following sentences. You may make up your own predicates.

   (13.15) *Abigail shares with Max.*

   (13.16) *Abigail reluctantly shares a toy with Max.*

   (13.17) *Abigail hates to share with Max.*

2. Find the PropBank framesets for *share* and *hate* at `http://verbs.colorado.edu/propbank/framesets-english-aliases/`, and rewrite your answers from the previous question, using the thematic roles ARG0, ARG1, and ARG2.

3. Compute the syntactic path features for Abigail and Max in each of the example sentences (13.15) and (13.17) in Question 1, with respect to the verb *share*. If you're not sure about the parse, you can try an online parser such as `http://nlp.stanford.edu:8080/parser/`.

4. Compute the dependency path features for Abigail and Max in each of the example sentences (13.15) and (13.17) in Question 1, with respect to the verb *share*. Again, if you're not sure about the parse, you can try an online parser such as `http://nlp.stanford.edu:8080/parser/`. As a hint, the dependency relation between *share* and *Max* is OBL according to the Universal Dependency treebank.

5. PropBank semantic role labeling includes **reference arguments**, such as,

   (13.18) [$_{\text{AM-LOC}}$ The bed] on [$_{\text{R-AM-LOC}}$ which] I slept broke.[15]

   The label R-AM-LOC indicates that the word *which* is a reference to *The bed*, which expresses the location of the event. Reference arguments must have referents: the tag R-AM-LOC can appear only when AM-LOC also appears in the sentence. Show how to express this as a linear constraint, specifically for the tag R-AM-LOC. Be sure to correctly handle the case in which neither AM-LOC nor R-AM-LOC appear in the sentence.

6. Explain how to express the constraints on semantic role labeling in Equation 13.8 and Equation 13.9 in the general form $\mathbf{A}\boldsymbol{z} \geq \boldsymbol{b}$.

7. Produce the AMR annotations for the following examples:

   (13.19) a. The girl likes the boy.

[15]Example from 2013 NAACL tutorial slides by Shumin Wu

b. The girl was liked by the boy.
c. Abigail likes Maxwell Aristotle.
d. The spy likes the attractive boy.
e. The girl doesn't like the boy.
f. The girl likes her dog.

For (13.19c), recall that multi-token names are created using `op1`, `op2`, etc. You will need to consult Banarescu et al. (2013) for (13.19e), and Schneider et al. (2015) for (13.19f). You may assume that *her* refers to *the girl* in this example.

8. In this problem, you will build a FrameNet sense classifier for the verb *can*, which can evoke two frames: POSSIBILITY (can you order a salad with french fries?) and CAPABILITY(can you eat a salad with chopsticks?).

   To build the dataset, access the FrameNet corpus in NLTK:

```
import nltk
nltk.download('framenet_v17')
from nltk.corpus import framenet as fn
```

   Next, find instances in which the lexical unit `can.v` (the verb form of *can*) evokes a frame. Do this by iterating over `fn.docs()`, and then over sentences, and then

```
for doc in fn.docs():
    if 'sentence' in doc:
        for sent in doc['sentence']:
            for anno_set in sent['annotationSet']:
                if 'luName' in anno_set and anno_set['luName'] == 'can.v':
                    pass # your code here
```

   Use the field `frameName` as a label, and build a set of features from the field `text`. Train a classifier to try to accurately predict the `frameName`, disregarding cases other than CAPABILITY and POSSIBILITY. Treat the first hundred instances as a training set, and the remaining instances as the test set. Can you do better than a classifier that simply selects the most common class?

9. *Download the PropBank sample data, using NLTK (`http://www.nltk.org/howto/propbank.html`).

   a) Use a deep learning toolkit such as PyTorch to train a BiLSTM sequence labeling model (§ 7.6) to identify words or phrases that are predicates, e.g., *we*/O *took*/B-PRED *a*/I-PRED *walk*/I-PRED *together*/O. Your model should compute the tag score from the BiLSTM hidden state $\psi(y_m) = \beta_y \cdot \boldsymbol{h}_m$.
   b) Optionally, implement Viterbi to improve the predictions of the model in the previous section.

c) Try to identify ARG0 and ARG1 for each predicate. You should again use the BiLSTM and BIO notation, but you may want to include the BiLSTM hidden state at the location of the predicate in your prediction model, e.g., $\psi(y_m) = \beta_y \cdot [\boldsymbol{h}_m; \boldsymbol{h}_{\hat{r}}]$, where $\hat{r}$ is the predicted location of the (first word of the) predicate.

10. Using an off-the-shelf PropBank SRL system,[16] build a simplified question answering system in the style of Shen and Lapata (2007). Specifically, your system should do the following:

    - For each document in a collection, it should apply the semantic role labeler, and should store the output as a tuple.
    - For a question, your system should again apply the semantic role labeler. If any of the roles are filled by a *wh*-pronoun, you should mark that role as the expected answer phrase (EAP).
    - To answer the question, search for a stored tuple which matches the question as well as possible (same predicate, no incompatible semantic roles, and as many matching roles as possible). Align the EAP against its role filler in the stored tuple, and return this as the answer.

    To evaluate your system, download a set of three news articles on the same topic, and write down five factoid questions that should be answerable from the articles. See if your system can answer these questions correctly. (If this problem is assigned to an entire class, you can build a large-scale test set and compare various approaches.)

[16] At the time of writing, the following systems are availabe: SENNA (`http://ronan.collobert.com/senna/`), Illinois Semantic Role Labeler (`https://cogcomp.cs.illinois.edu/page/software_view/SRL`), and mate-tools (`https://code.google.com/archive/p/mate-tools/`).