Introduction to Semantic Role Labelling

Behrang QasemiZadeh
zadeh@phil.hhu.de
Computational Linguistics Department, HHU – DRAFT

October 2018–January 2019

Recap

What we did:

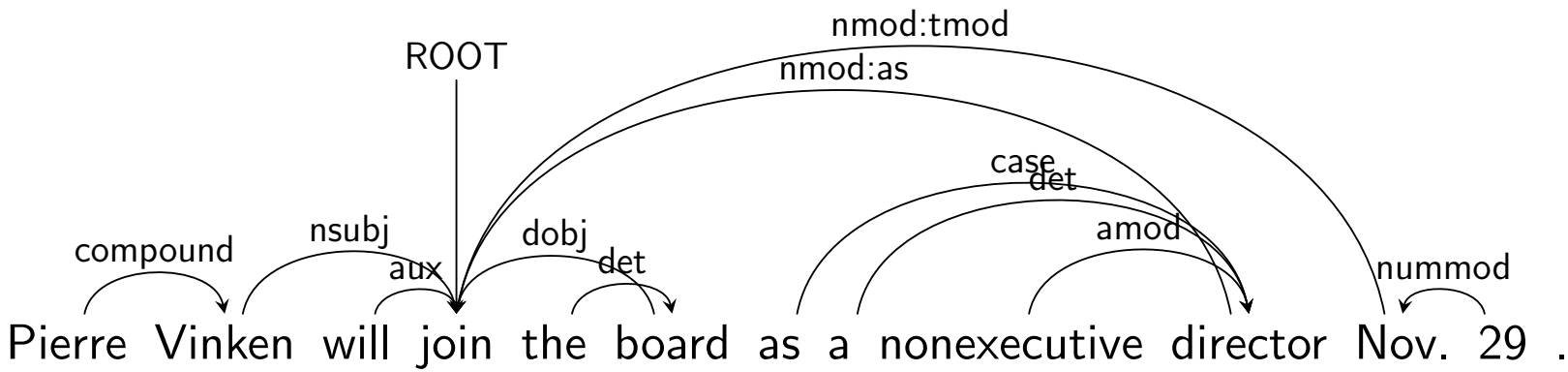
+ spent a good amount of time discussing the idea behind record-by-feature representation (describing things by things).
+ understood that choosing the records (what we represent) and features (how we represent) has an immediate effect on the outcome of our classification task.
+ discussed, informally, how these record-by-features can be used in classification.
+ examined the syntactic dependency parse of a sentence in detail.
+ examined semantic role representation of the same sentence in the SDP format.

Recap: Syntactic Dependencies

Given the sentence

> *Pierre Vinken, 61 years old, will join the board as a nonexecutive director Nov. 29.*

its syntactic dependency parse is



Recap: Syntactic Dependencies (contd.)

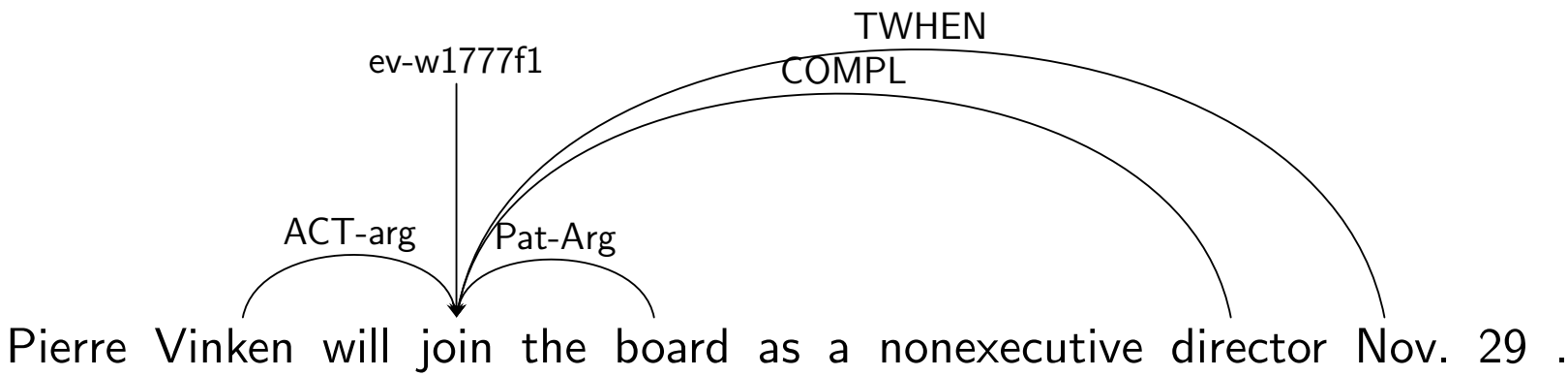
which is available for processing in the CoNLL-U format:

1	Pierre	Pierre	PROPN	NNP	_	2	compound	_	_
2	Vinken	Vinken	PROPN	NNP	_	9	nsubj	_	_
3	,	,	PUNCT	,	_	2	punct	_	_
4	61	61	NUM	CD	_	5	nummod	_	_
5	years	year	NOUN	NNS	_	6	nmod:npmod	_	_
6	old	old	ADJ	JJ	_	2	amod	_	_
7	,	,	PUNCT	,	_	2	punct	_	_
8	will	will	AUX	MD	_	9	aux	_	_
9	join	join	VERB	VB	_	0	root	_	_
10	the	the	DET	DT	_	11	det	_	_
11	board	board	NOUN	NN	_	9	dobj	_	_
12	as	as	ADP	IN	_	15	case	_	_
13	a	a	DET	DT	_	15	det	_	_
14	nonexecutive	nonexecutive	ADJ	JJ	_	15	amod	_	_
15	director	director	NOUN	NN	_	9	nmod	_	_
16	Nov.	Nov.	PROPN	NNP	_	9	nmod:tmod	_	_
17	29	29	NUM	CD	_	16	nummod	_	_
18	.	.	PUNCT	.	_	9	punct	_	_

Recap: Semantic Role Dependencies

The SDP-PSD corpus annotated syntactic dependencies with semantic role. For instance, for the verb *join*:

JOIN.EV-W1777F1	
ACT-ARG	*Vinken*
PAT-ARG	*board*
COMPL	*director*
TWHEN	*Nov.*



Recap: Semantic Role Dependencies (contd.)

which in turn is represented and available for processing in a TSV file format exemplified below:

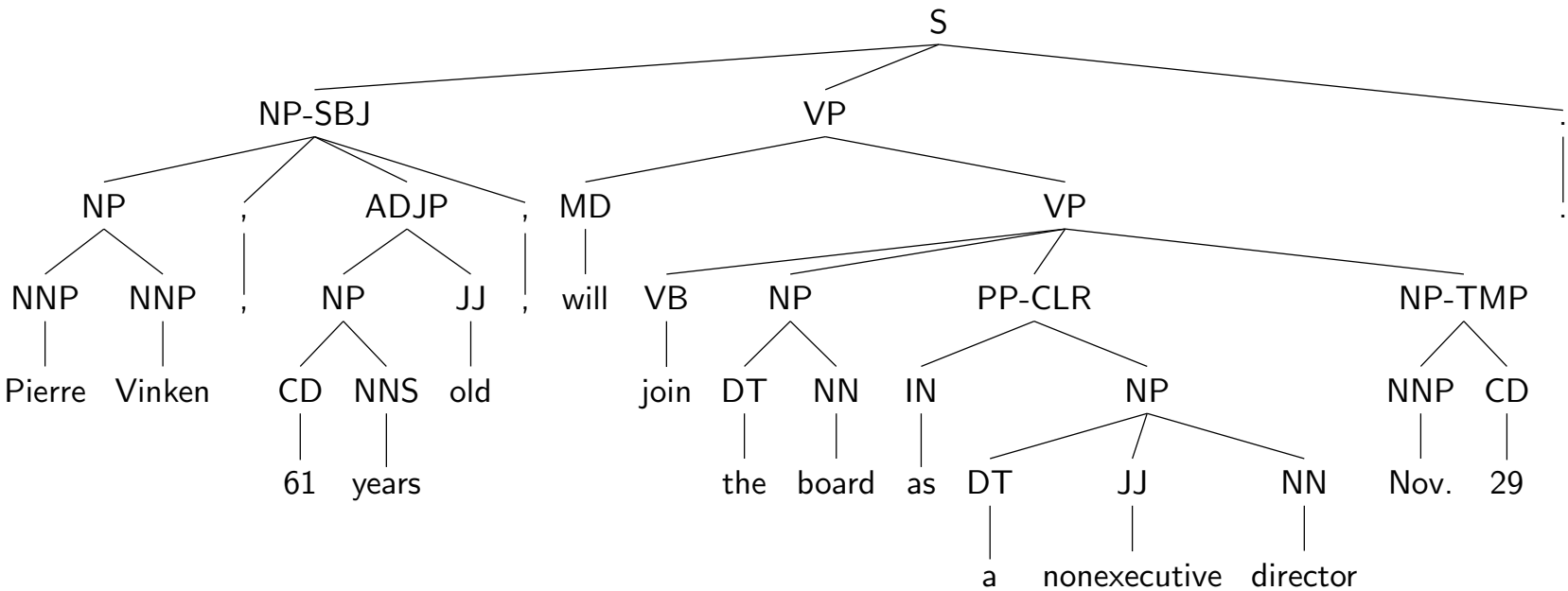
1	Pierre	Pierre	NNP	-	-	_		NE	_	_	_	_	_
2	Vinken	vinken	NNP	-	+	_		_	_	_	ACT-arg	_	_
3	,	,	,	-	-	_		_	_	_	_	_	_
4	61	61	CD	-	-	_		_	RSTR	_	_	_	_
5	years	year	NNS	-	+	_		_	_	EXT	_	_	_
6	old	old	JJ	-	+	_		DESCR	_	_	_	_	_
7	,	,	,	-	-	_		_	_	_	_	_	_
8	will	will	MD	-	-	_		_	_	_	_	_	_
9	join	join	VB	+	+	ev-w1777f1		_	_	_	_	_	_
10	the	the	DT	-	-	_		_	_	_	_	_	_
11	board	board	NN	-	-	_		_	_	_	PAT-arg	_	_
12	as	as	IN	-	-	_		_	_	_	_	_	_
13	a	a	DT	-	-	_		_	_	_	_	_	_
14	nonexecutive	nonexecutive	JJ	-	-	_		_	_	_	_	RSTR	_
15	director	director	NN	-	+	_		_	_	_	COMPL	_	_
16	Nov.	nov.	NNP	-	+	_		_	_	_	TWHEN	_	_
17	29	29	CD	-	-	_		_	_	_	_	_	RSTR
18	.	.	.	-	-	_		_	_	_	_	_	_

Our Goal Today ...

Today, our focus is on basic feature representations for the Semantic Role Labelling Task.

We go through Chapter 3 of Palmer et al. (2010) and in particular look at syntactic features based on constituent parses.

Constituent-based Syntactic Parsing



which is often represented using a bracketed text format:
(S (NP-SBJ (NP (NNP Pierre) (NNP Vinken)) (, ,) (ADJP (NP (CD 61) (NNS years)) (JJ old)) (, ,)) (VP (MD will) (VP (VB join) (NP (DT the) (NN board)) (PP-CLR (IN as) (NP (DT a) (JJ nonexecutive) (NN director))) (NP-TMP (NNP Nov.) (CD 29)))) (. .))

Constituent-based Syntactic Parsing (contd.)

```
(S
  (NP-SBJ
    (NP (NNP Pierre) (NNP Vinken))
    (, ,)
    (ADJP (NP (CD 61) (NNS years)) (JJ old))
    (, ,))
  (VP
    (MD will)
    (VP
      (VB join)
      (NP (DT the) (NN board))
      (PP-CLR (IN as) (NP (DT a) (JJ nonexecutive) (NN director)))
      (NP-TMP (NNP Nov.) (CD 29))))
  (. .))
```

Semantic Role Labeling as a Classification Task

+ Semantic Role Labeling can be modeled as a classification task.
+ In its simplest form, for a given verb, and given each syntactic constituent in a parse tree, a semantic role labeler decides:
  - Whether a constituent is an argument/adjunct of the verb?
  - If yes, what is the semantic role label for the constituent?

  Note that
  - In a basic approach, the two questions above are merged into one classification problem:Simply add new class labels to the semantic role labels.
  - The labels are predefined (classification task): these are mostly the labels that are asserted in the training corpus (e.g., for PropBank these are `Arg0`, `Arg1`, and so on).

The Basic Mechanism for Semantic Role Labeling

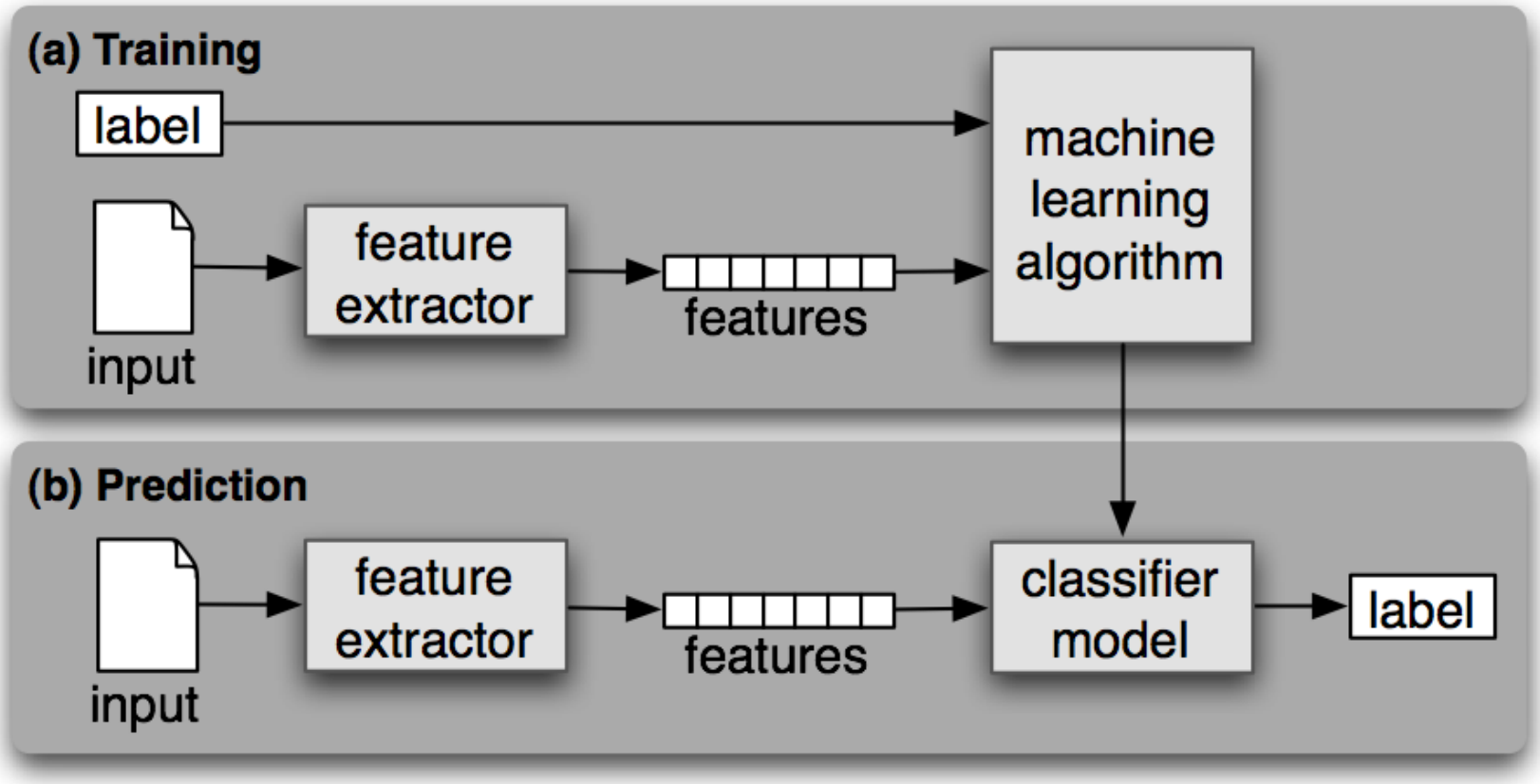
As you guess, the method is simple:

Represent each constituent with a record-by-feature representation through the so-called feature design/extraction[1] process.

Train a standard machine learning algorithm.

Use the learned model to predict the semantic role labels.

The Basic Mechanism for Semantic Role Labeling (contd.)



[2]

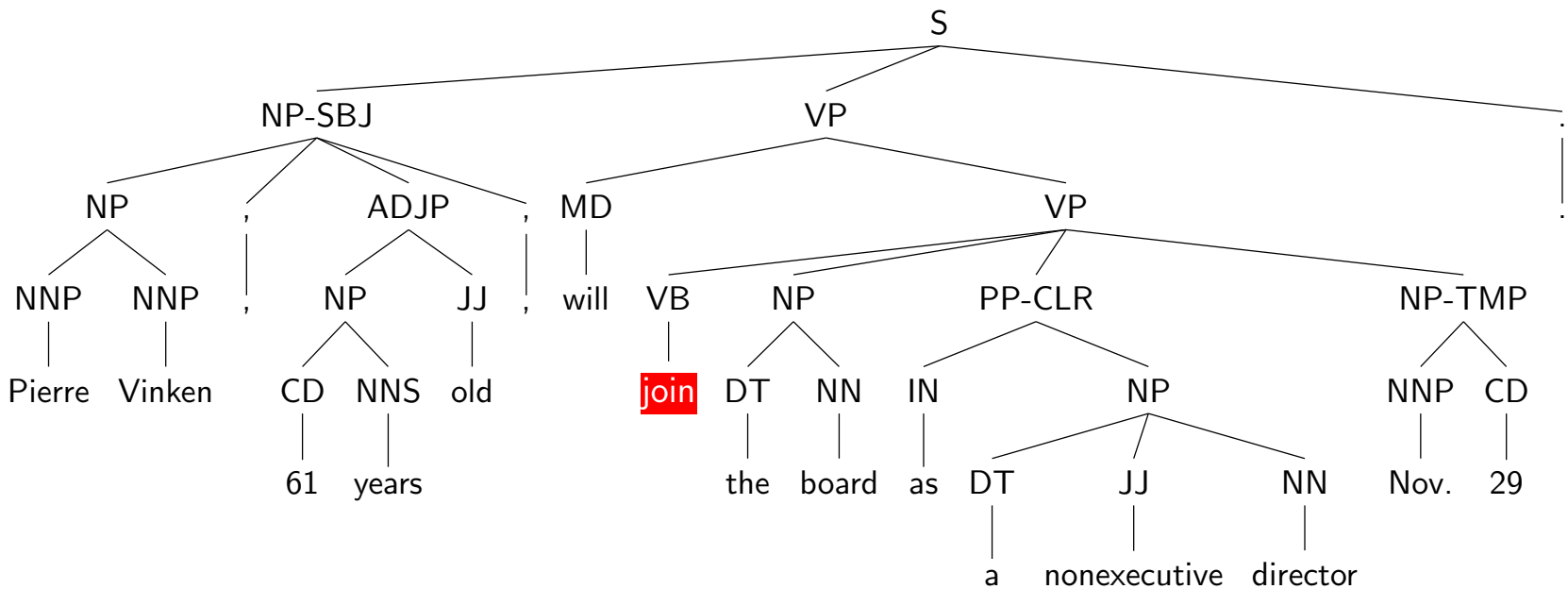
See Introduction to Classification in (Bird et al., 2009, Chap. 6).

[1] This feature extraction is not used in the sense that is used in machine learning literature.

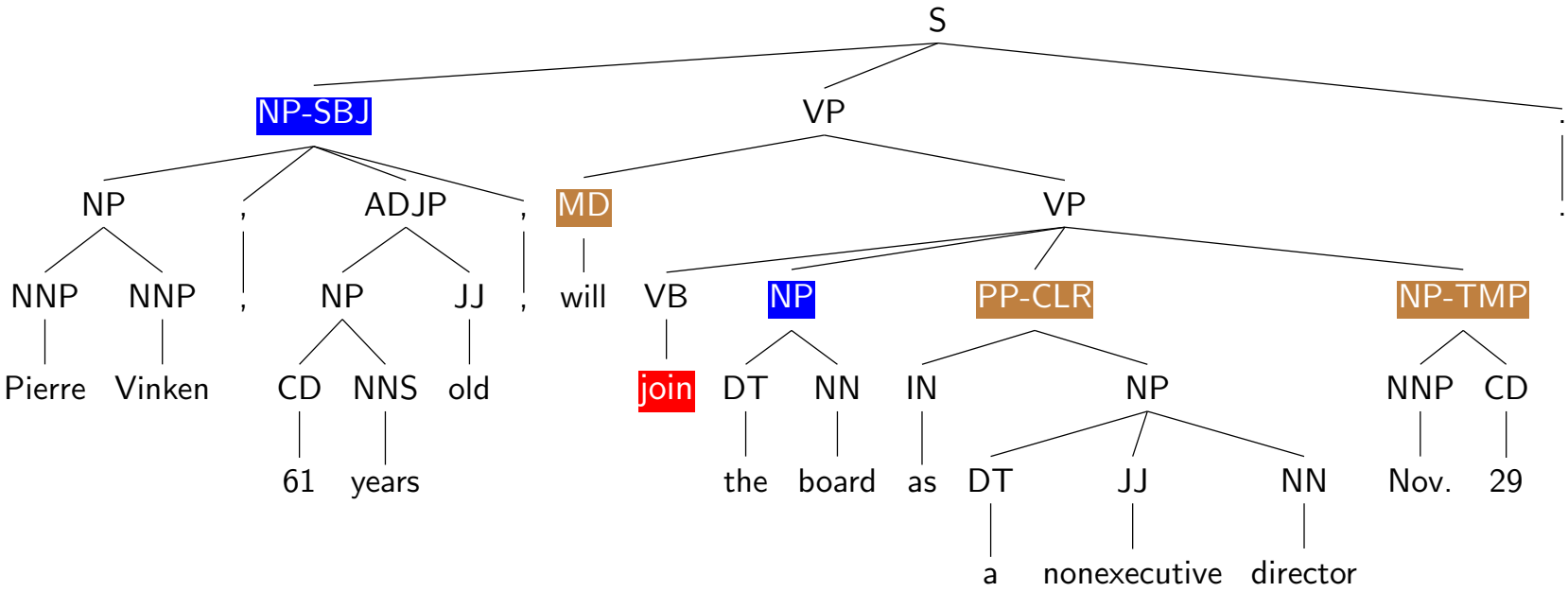
[2] Source: `https://www.nltk.org/book/ch06.html`

Semantic Role Labeling as a Classification Task
Identification Problem

Assume we want to find semantic roles with respect to join: How many constituents do you recognize? Which one of them are your candidates for representation?



Semantic Role Labeling as a Classification Task
Identification Problem (contd.)



Given the recursive nature of constituent parses, we usually have a large number of constituents in a parse tree.

But, only a small number of them (in blue/brown) are relevant to the target verb.

Semantic Role Labeling as a Classification Task
Identification Problem (contd.)

Apart from a large number of constituents in a parse tree, it is possible that a constituents to be assigned to different semantic role by different predicates.

Not only that, even the same verb/predicate can assign multiple roles (e.g. in FrameNet) to a constituent (multiple labels for one record).

Semantic Role Labeling as a Classification Task Identification Problem (contd.)

Simple methods such as presenting "one record per constituent per verb" would not work in practice due to the imbalanced portion of negative and positive samples in the resulting training data:

- Most records will be negative samples (of no semantic relation to the predicate).
- The highly imbalanced/skewed data causes a machine learning algorithm to assign all records to one label (the most frequent one).

To solve this problem, we often break down the semantic role labelling task as an Identification and label-classification subtasks.

Semantic Role Labeling as a Classification Task
Identification Problem (contd.)

Identification sub-task is a binary classification task:

+ Given a constituent and a verb, decide whether they are related or not.
+ The binary classifier acts like a filter.
+ This is where you can use syntactic Parsing techniques to improve results.

    - Remember that the decisions made about records are not entirely independent (we get back to this later).

Semantic Role Labeling as a Classification Task
Identification Problem (contd.)

In the second step, the label-classification deals only with the output of the Identification step. Often this improves the result.

Often, both classifiers use the same feature representation.

However, simple heuristics can replace the identification task, too:

> Use rule of thumbs or more sophisticated constraints to filter negative samples.

Or, maybe combine the two: (a) Remove noise (negative samples) as much as it does not hurt the system's recall; and then (b) train an Identifier on this data.

Semantic Role Labeling as a Classification Task
Identification Problem (contd.)

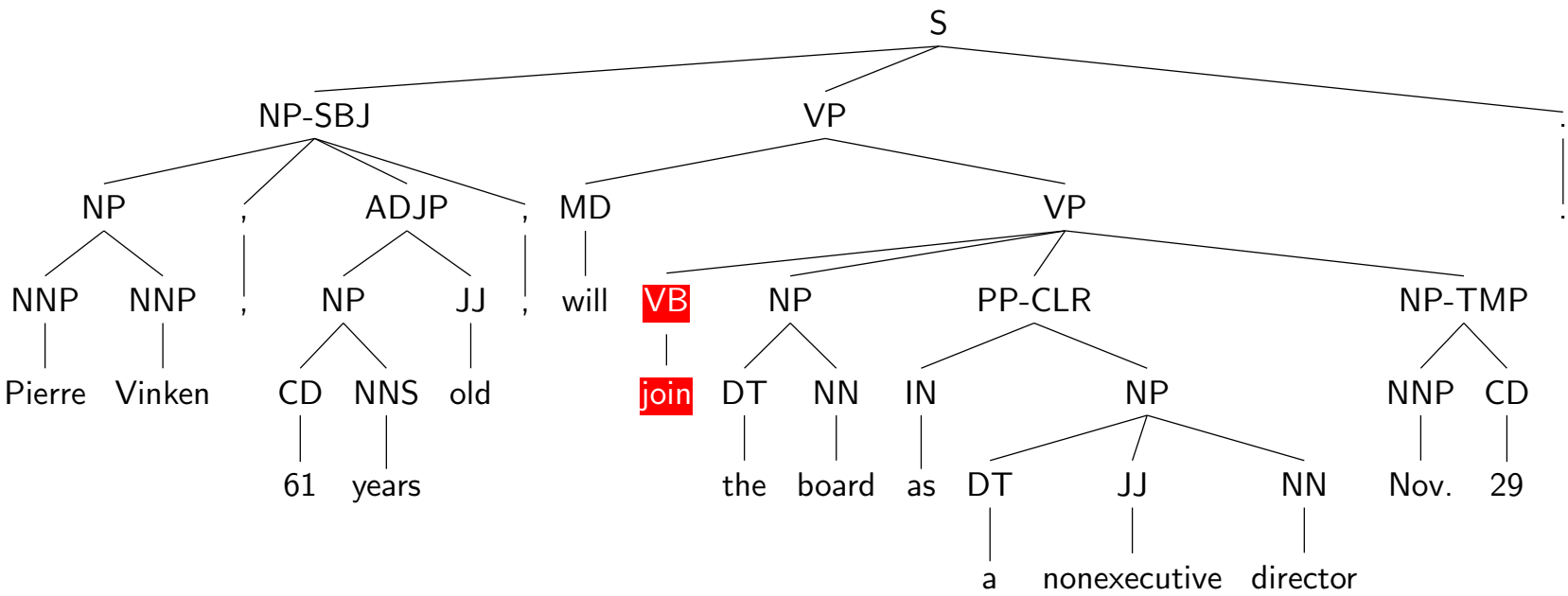
Xue and Palmer (2004) is an effective filtering method:

> Initialize an empty list of candidates $\mathcal{L}$
>
> Add all the sister nodes of the target verb to the $\mathcal{L}$
>
> Go up one level and do a similar thing unless for conjunctions.

Let's look at an example.

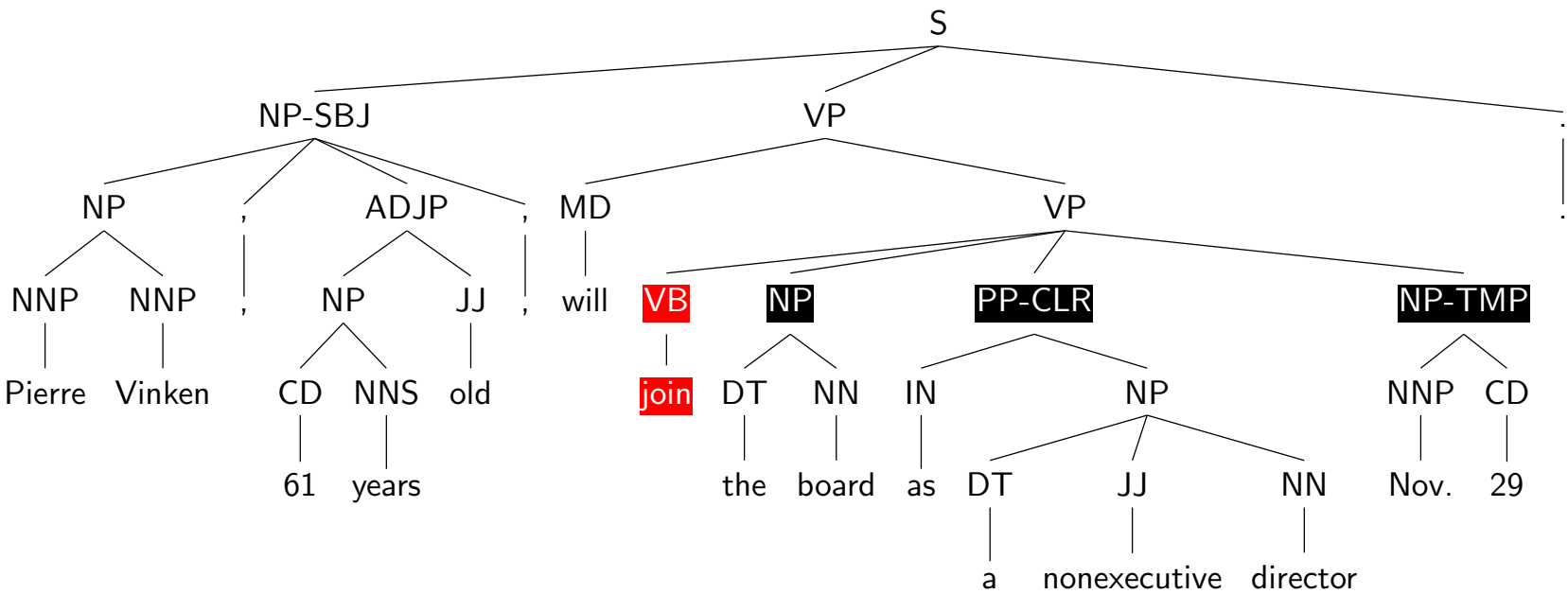
Semantic Role Labeling as a Classification Task
Identification Problem (contd.)

The target verb is VB(join)



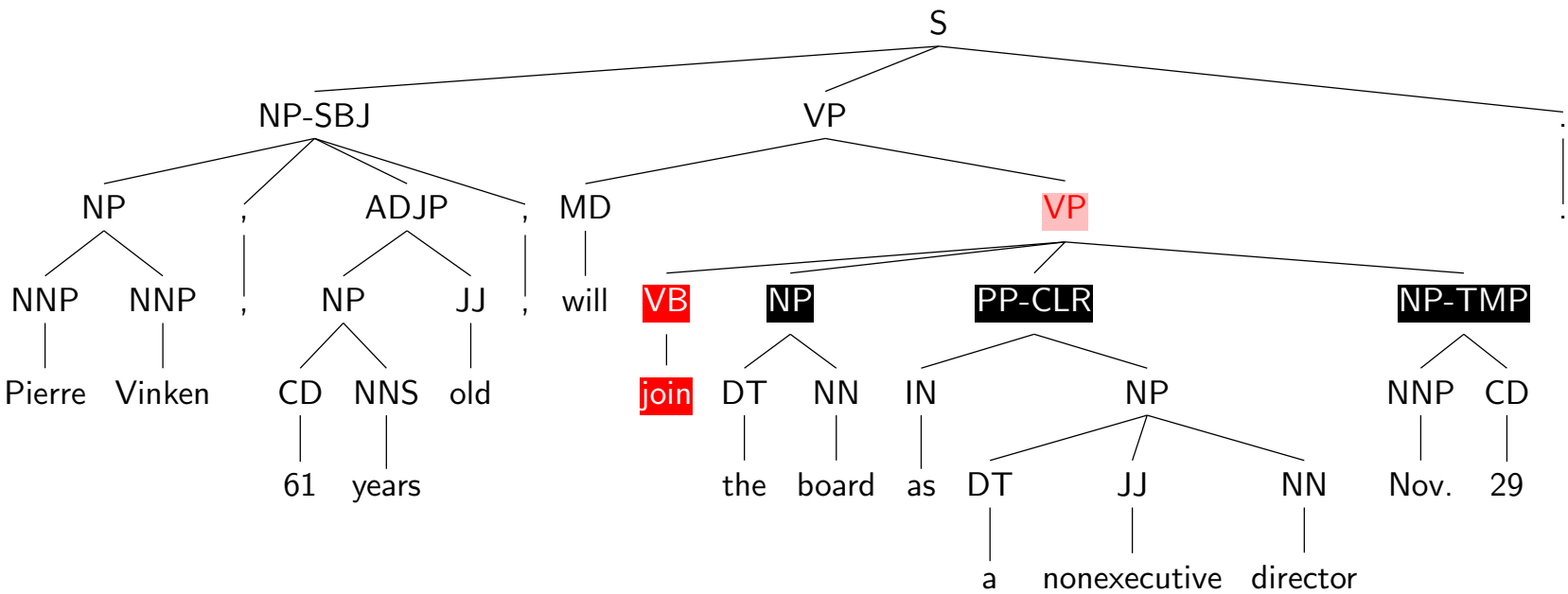
Semantic Role Labeling as a Classification Task
Identification Problem (contd.)

Add the sister nodes of VB(join)



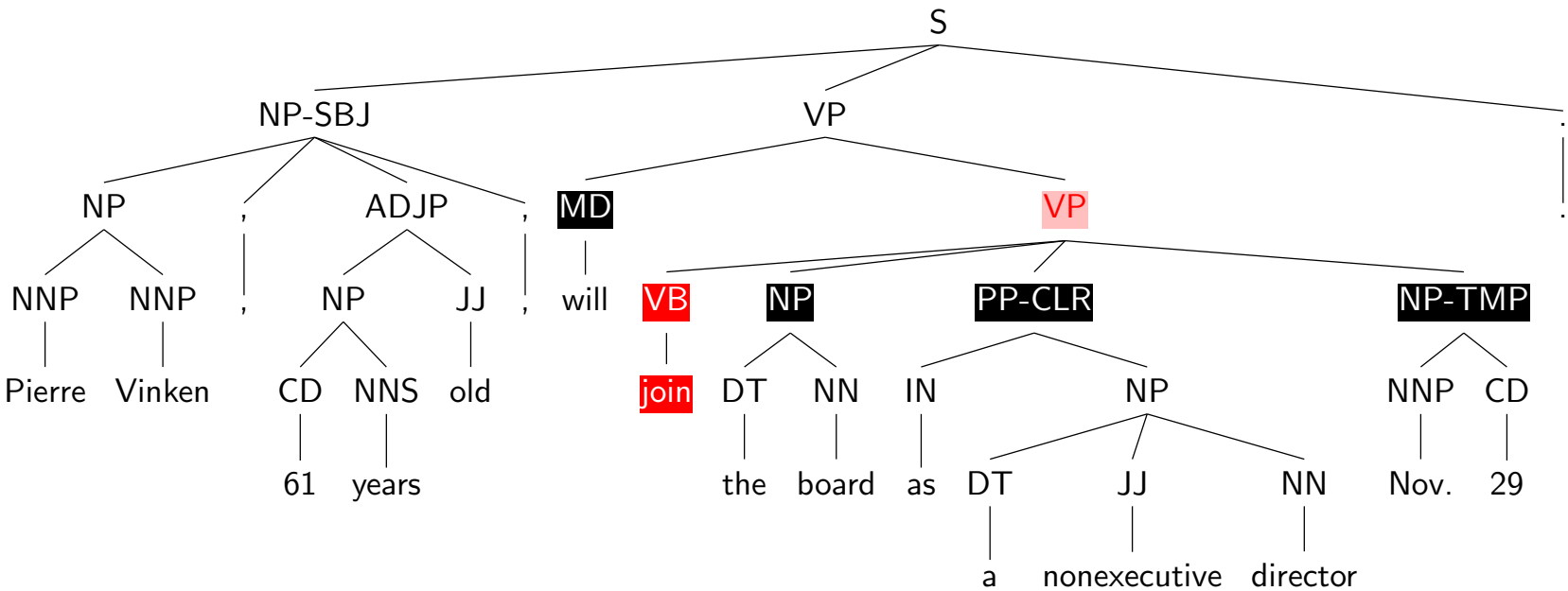
Semantic Role Labeling as a Classification Task
Identification Problem (contd.)

Go up from VB(join) to VP



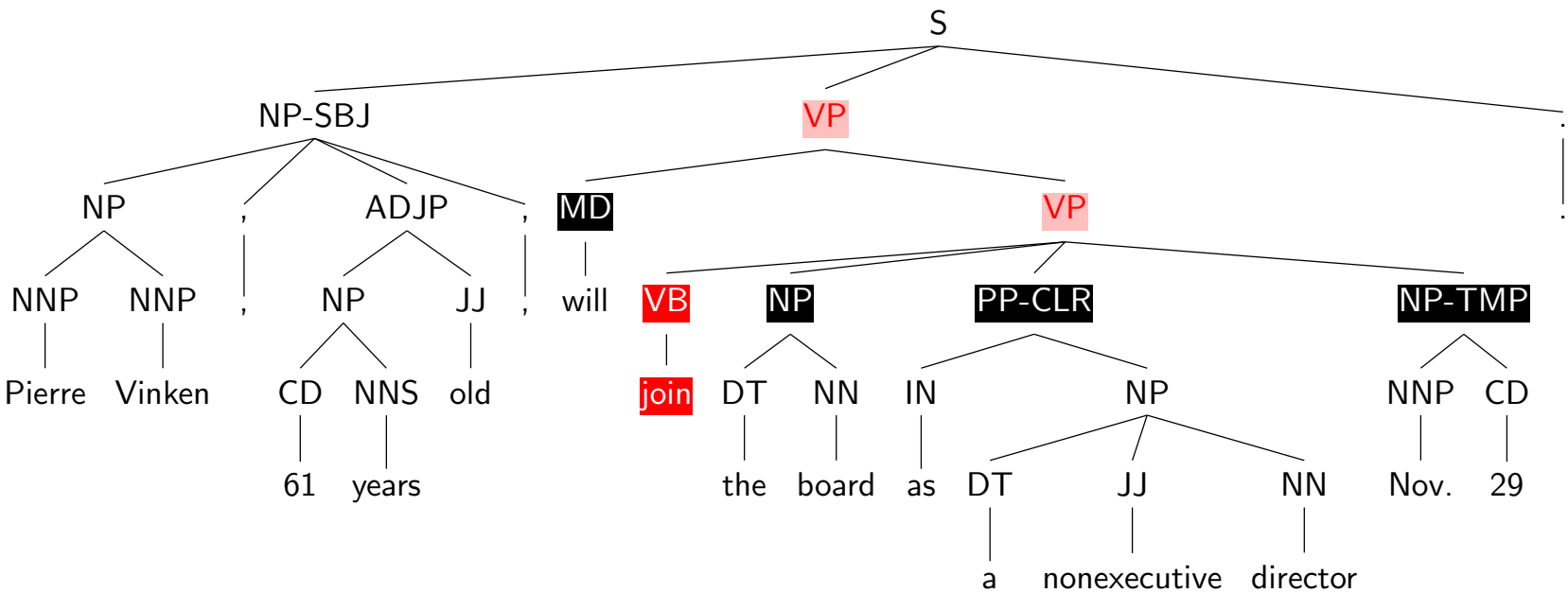
Semantic Role Labeling as a Classification Task
Identification Problem (contd.)

Add all the sisters of VP



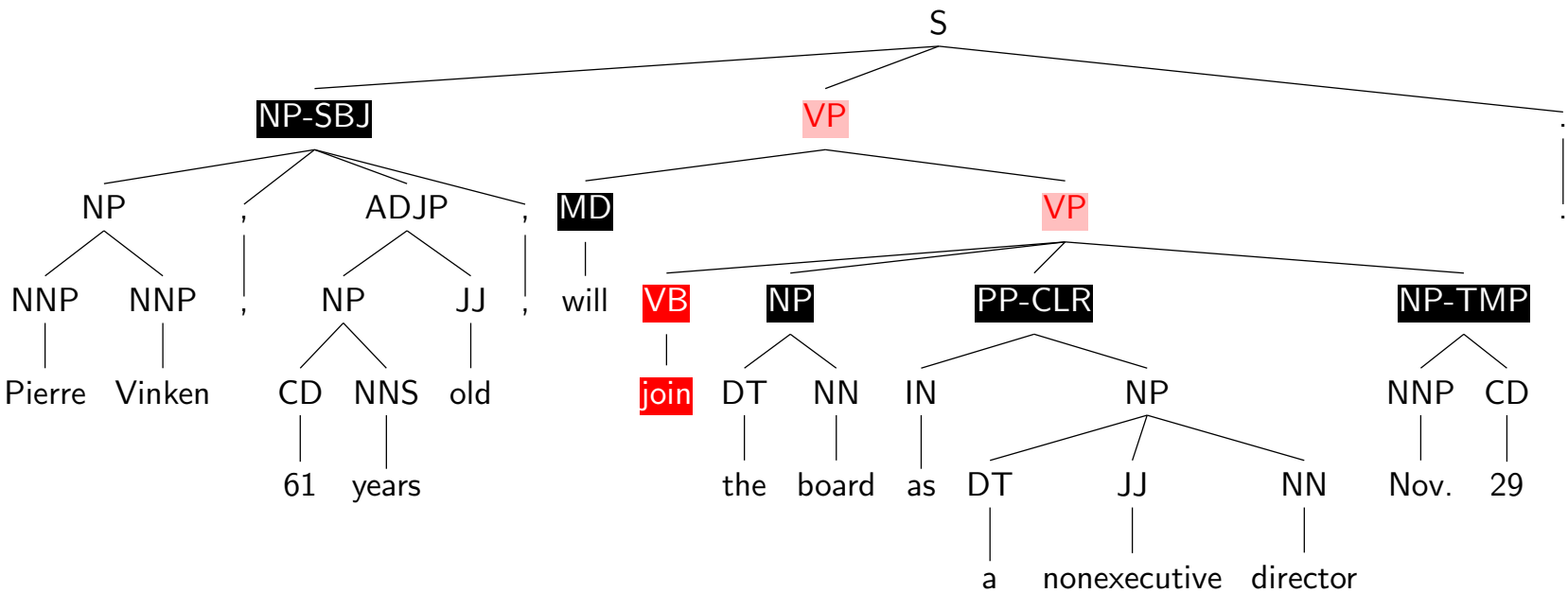
Semantic Role Labeling as a Classification Task
Identification Problem (contd.)

Go up from VP to its parent VP.



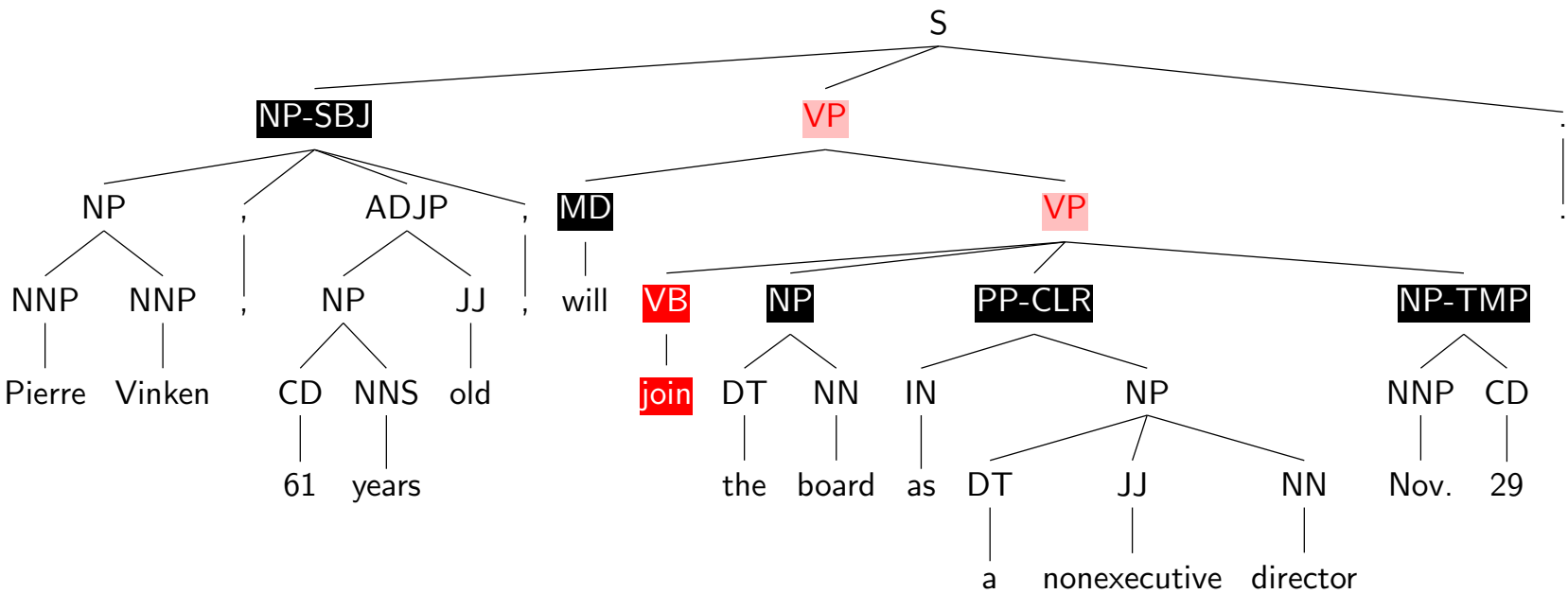
Semantic Role Labeling as a Classification Task
Identification Problem (contd.)

Add all the sisters.



Semantic Role Labeling as a Classification Task
Identification Problem (contd.)

No way to go up; done! All the nodes in black are candidates $\mathcal{L}$.



Semantic Role Labeling as a Classification Task Identification Problem (contd.)

It happens that the method of Xue and Palmer (2004) gives a prefect $\mathcal{L}$ for the verb `join` in our example.

To continue, we must build feature-representation for $l \in \mathcal{L}$.

For training a classifier, we must also assign $l \in \mathcal{L}$ to their class labels – i.e., what is what.

Let's start with assigning PropBank style class labels to our *l*s.

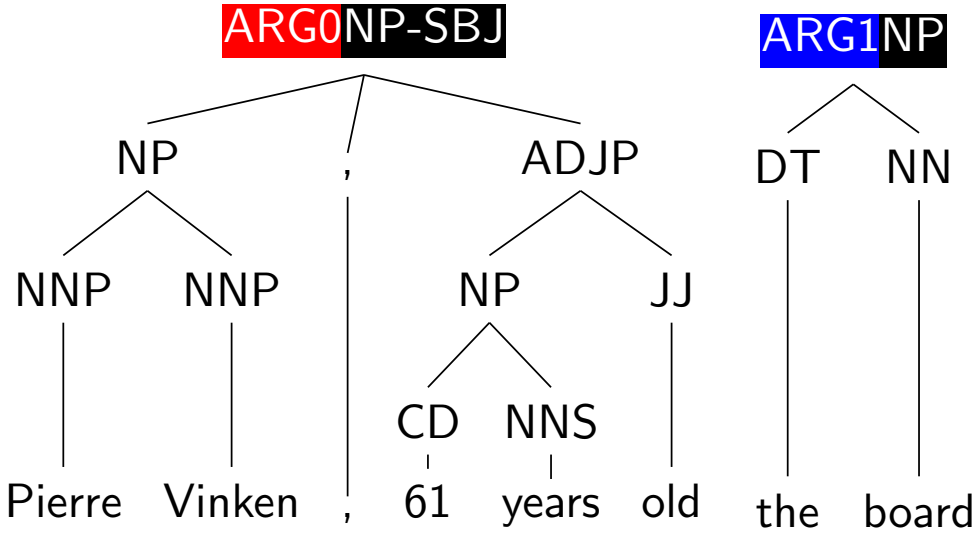
Preparing training data: Assigning labels

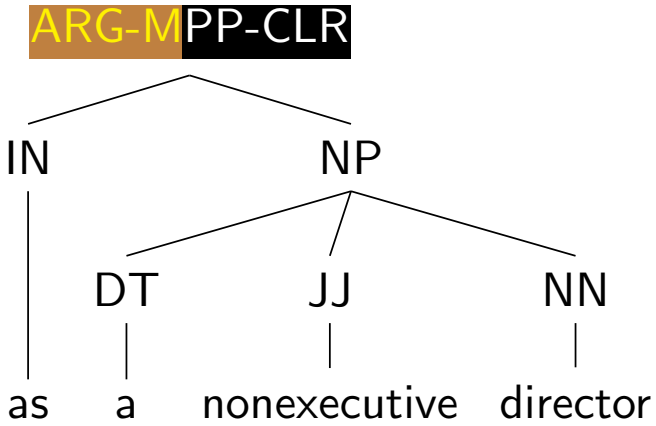
To get the class labels, we need to align $\mathcal{L}$ with our annotations from PropBank:

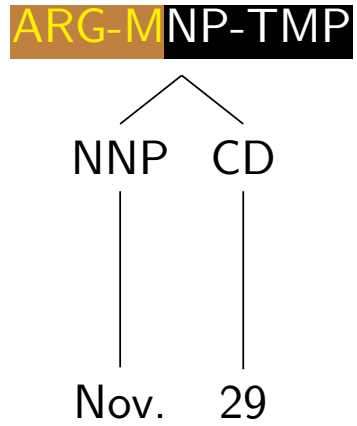
[ARG0 Pierre Vinken , 61 years old ,] [ARGM-MOD will] [rel join] [ARG1 the board] [ARGM-PRD as a nonexecutive director] [ARGM-TMP Nov. 29] .

Obviously we assume our sentences (for training and testing a classifier) are previously annotated.

Preparing training data: Assigning labels (contd.)

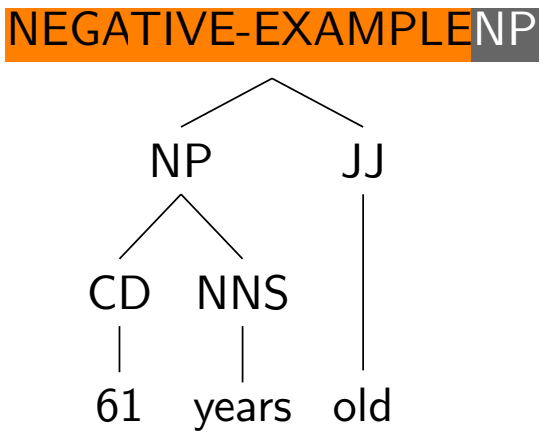






Preparing training data: Assigning labels (contd.)

If $\mathcal{L}$ contains constituents which are not a semantic dependant of the verb, we can simply label them as "NEGATIVE-EXAMPLE". For instance:



NB: A balanced "high-quality" set of negative samples are as important as positive examples.

Preparing training data: Assigning labels (contd.)

The next step is to describe each of the labelled constituents with respect to a set of features.

Feature representation of candidates

Please be reminded that:

- There is no fixed set of universal features recommended for Semantic Role Labelling systems.
- There is no universal methods for selecting and identifying relevant features.
- There is no fixed set of guidelines on how to encode features for a particular learning method.

Feature representation of candidates (contd.)

But, there are some best practices.

We can often obtain a reasonable performance by using a relatively simple set of features, such as those suggested in Palmer et al. (2010).

However, the performance can be improved significantly by altering the employed features, often through a trial-and-error.

We often use a *greedy* method: We define as many features as we can. Then we use the so called kitchen sink method Bird et al. (2009) to filter irrelevant features.

Otherwise, instead of kitchen sink method we can also use neural network.

Feature representation of candidates (contd.)

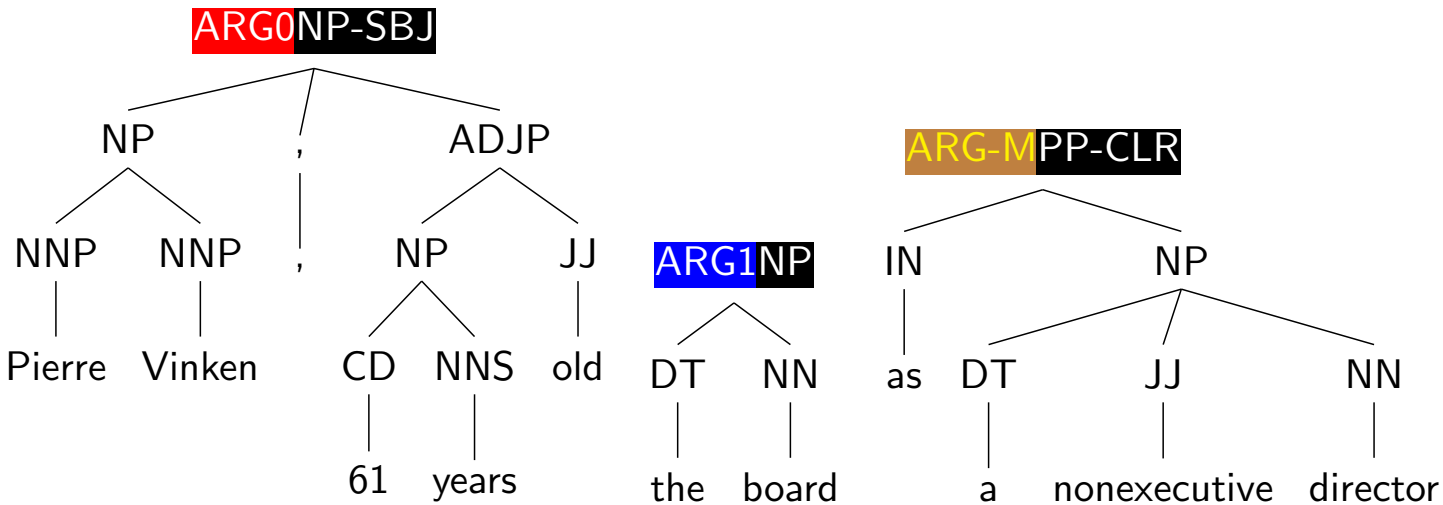
In any case, first we need to come up with a (a) feature representation and then (b) encoding for the constituents in $\mathcal{L}$.

An elaborate summary of classic features used in SRL is provided in our text book Palmer et al. (2010).

PLEASE READ PP. 33–42 and ask your questions the next session. These pages are appended at the end of these slides, no excuse to bypass them.

Feature representation of candidates (contd.)

Phrase type: syntactic categories of $l \in \mathcal{L}$ are important. So we often use syntactic parsers both for identifying candidate constituents as well as determining their syntactic categories (such as depicted in the previous example): **NP-SBJ, NP, PP-CLR, . . .**



NB: Syntactic parsers produce a variety of output. Not all of them provide information such as above.

Feature representation of candidates (contd.)

Some parsers yield more informative output by marking subject, direct object, complements, etc. Others would only give basic information such NP, VP, PP, and so on.

It is expected that parses of more refined syntactic categories contain more errors that the general ones (though not necessarily true always).

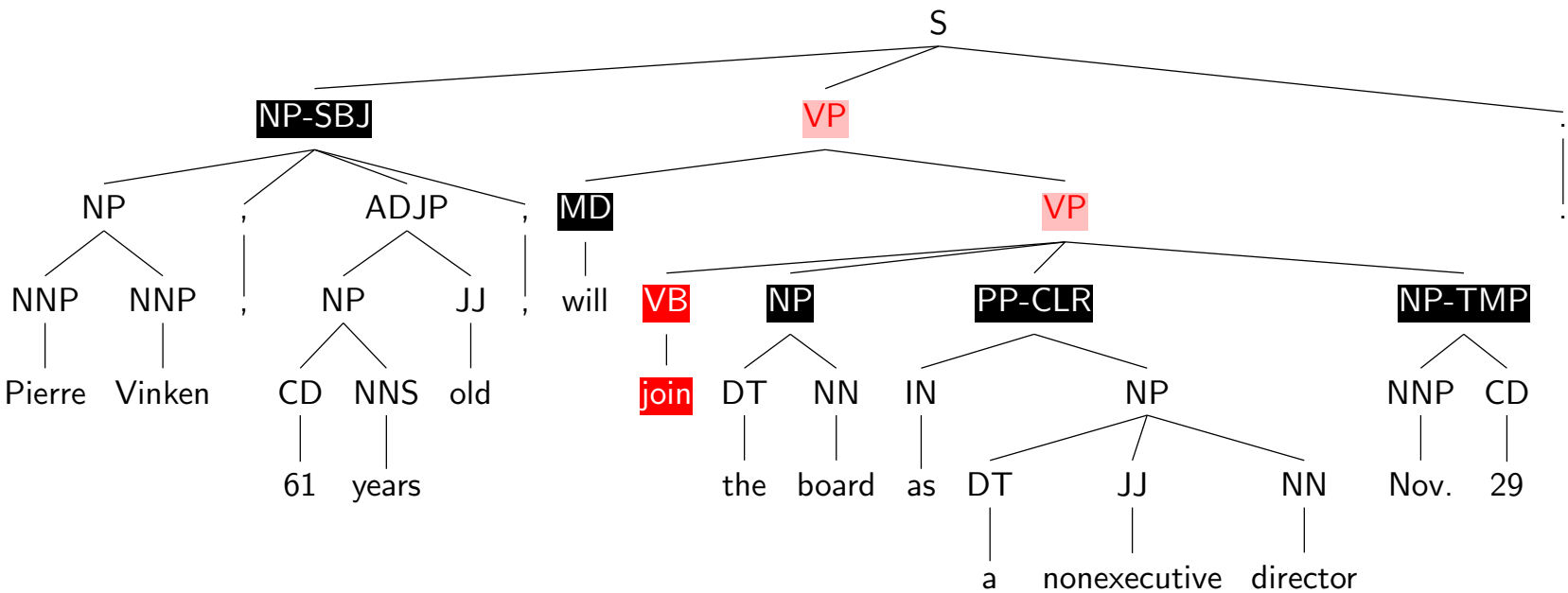
Constituent parses give a hierarchically structured output which are potentially more informative of dependency parses that assert pairwise relations between words.

Caveat: Errors in the automatic parses.

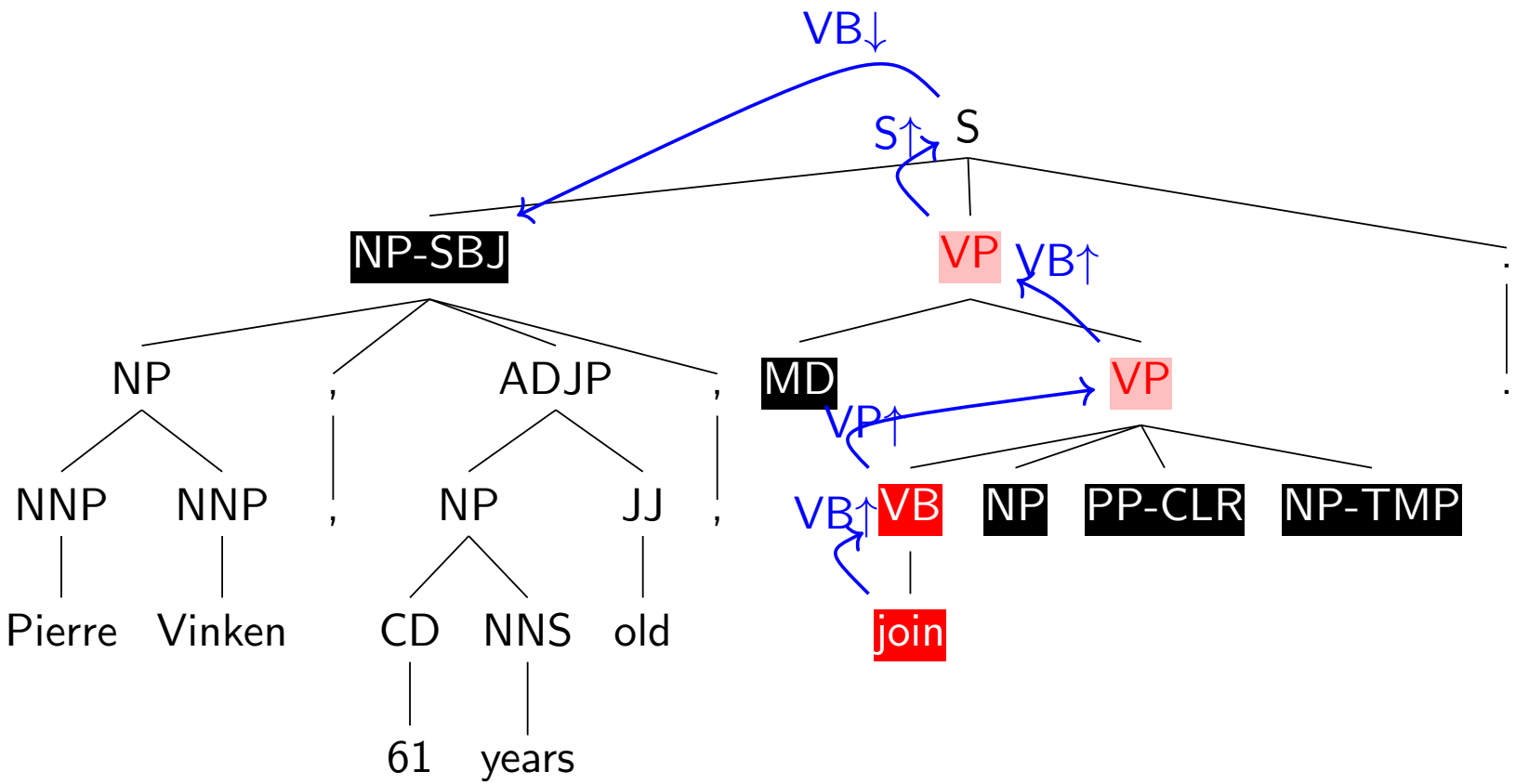
Feature representation of candidates (contd.)

Parse Tree Path: Describes the syntactic relation between the target verb and candidate constituents.

For instance, look at the path between join and the NP-SBJ (ARG0) node.



Feature representation of candidates (contd.)



which we can simply write as: VB↑VP↑VP↑S↓NP-SBJ

Feature representation of candidates (contd.)

Just to make sure that I had been clear:

- Path features can be defined in an arbitrary number of ways;
- Do not define paths that are too specific, also avoid defining those that are too generic;
- Syntactic formalism affects the path definitions;
- Most often we use paths defined over dependency parses, too;
- Paths can be also used to describe relations between candidate roles (seen later).

Feature representation of candidates (contd.)

Governing Category: From a candidate constituent go up until you reach a S or VP node; note this node category ("S" or "VP") as the value for the *Governing Category* feature.

The feature is applied only to candidate constituents of "NP" category: NP nodes under a S node are usually syntactic subjects; NP nodes under VP are usually syntactic objects.

Given the locality of paths for this feature, they are robuster than *path features*.

Feature representation of candidates (contd.)

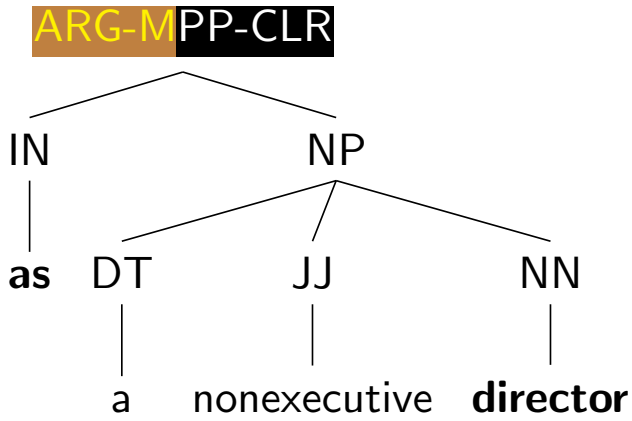
Position: relative position of the target verb and the candidates (e.g., before or after, how many tokens, etc.).

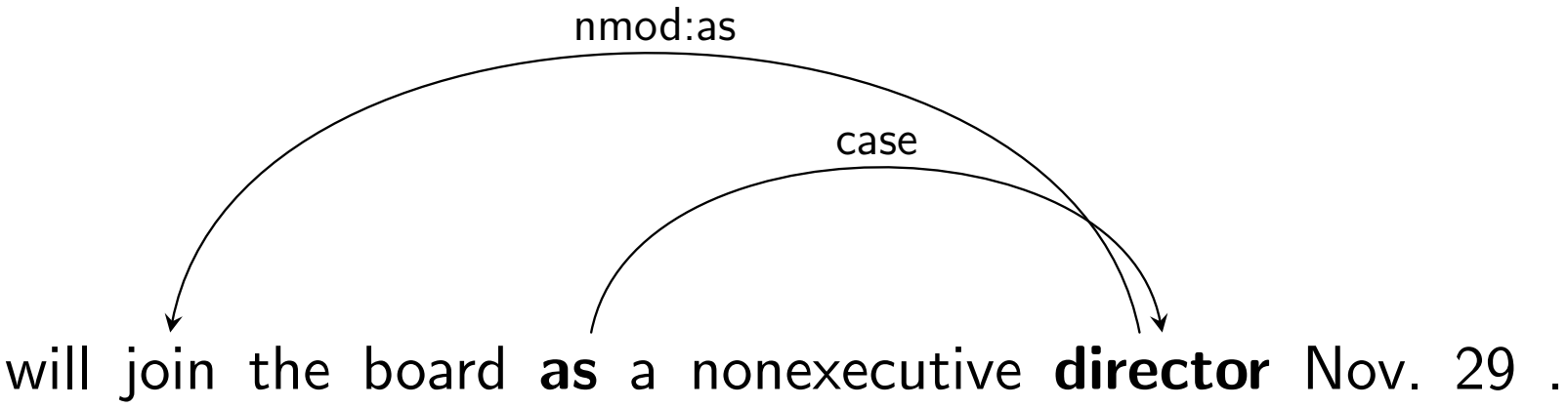
Voice: whether the target verb is in active or passive voice;

Head Words + Prepositions: the lexical "head" of candidate constituents (these are usually content words), e.g., the preposition *as* and the word *director* in the constituent *as a nonexecutive director*.

For extracting them, you can either use path features over constituents, or syntactic dependency parses.

Feature representation of candidates (contd.)





Feature representation of candidates (contd.)

Subcategorization frames: The set of syntactic arguments of the verb in the sentence; in our sentence, this can be defined/extracted (ideally) as {subject, object}, or {NP, NP, PP,NP}, . . . (depending on what you get from your parser).

This feature informs the system on what is happening in the broader context of candidate constituents.

Argument Set: The set of all roles appeared for the verb in the sentence (in the above, replace syntactic categories with semantic role labels).

Feature representation of candidates (contd.)

This Argument Set feature is quiet interesting: During the training, we can produce these *Argument Set* features. What about the prediction cycle?

Can you recognize the dynamic that its use creates in the system?

This can be implemented in a variety of ways:

- Simply use the sequence of automatically generated labels so far and use it as feature for the next ones;
- Impose additional constraints on the labels relationships, e.g., use dynamic programming/Bayesian methods/....

Feature representation of candidates (contd.)

There are many other features to be used:

Argument Order, Previous Role, Head Word Part-of-Speech Category, Named Entities in Constituents (Person, Organization, Time, Date, etc.), Dictionary-based Features e.g. Verb Classes (such as Levin Classes/VerbNet), Verb Clustering (e.g., use Brown clustering), First and Last Word/POS in Constituents, Constituent Order, Constituent Tree Distance, Constituent Context Features (parent, left and right siblings of the constituent), Cue Words, ...

If you have parallel text, perhaps you would consider using cross-lingual features.

If you are using a modern classifier, then a range of embedding techniques can be used too (e.g., see He et al. (2017)).

Practice Today's End Goal . . .

Hopefully, now and after this introduction, you can simply come up with a feature representation for the candidate constituents that we had, e.g.:

Class Label	**Feat-Category**	**Feat-HeadWord**	**Feat-Position**	**Feat-SubCat**	**. . .**
ARG0	NP	Vinken	Left	{N,N,P,N}	. . .
NEGATIVE	NP	years	Right	{N,N,P,N}	. . .
. . .	. . .	. . .	. . .	. . .	. . .

Could you create a feature table such as above for the 5 candidate constituents in our example?

We are still a small step behind using a classifier: This features must be organized in a structure suitable for the intended classification algorithm.

Bibliography

Bird, S., Klein, E., and Loper, E. (2009). *Natural Language Processing with Python*. O'Reilly Media, Inc., 1st edition.

He, L., Lee, K., Lewis, M., and Zettlemoyer, L. (2017). Deep semantic role labeling: What works and what's next. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 473–483. Association for Computational Linguistics.

Palmer, M., Gildea, D., and Xue, N. (2010). *Semantic Role Labeling*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Xue, N. and Palmer, M. (2004). Calibrating features for semantic role labeling. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*.

A heuristic algorithm that has been widely adopted is the one first proposed in Xue and Palmer (2004). The algorithm starts by locating the predicate for which the semantic role labeling system is identifying arguments, and then adds its sisters in the tree to the list of candidates that are potential arguments for this predicate. It then iteratively goes up a level and adds the sisters to its parent, grandparent, and so forth to the list of candidates until it reaches the root node of the tree. At each level, it first tries to decide whether there is a coordination structure. Conjuncts in a coordination structure are not possible arguments and are thus excluded from the list of candidates.

This process is illustrated in (20). Assuming that the predicate of interest is "warned," the system first adds the PP "of tough measures" to the list of candidates. It then moves up a level and adds the NP "Premier Ryzhkov" to the list of candidates. At the next level, the two S's form a coordination structure, and thus no candidate is added.

The pruning algorithm is more accurate when the parse trees that are the input to the semantic role labeling system are correct. In a realistic scenario, the parse trees are generated by a syntactic parser and are not expected to be perfect. However, experimental results show that even when the parses are imperfect, using a pruning algorithm leads to an improvement in the overall semantic role labeling accuracy.

3.2 FEATURES USED FOR CLASSIFICATION

The set of features used in SRL systems has grown over time as researchers have explored new ways of leveraging the syntactic analysis of the entire sentence to better analyze specific semantic roles. Thus, while early systems used only a handful of features, current state of the art systems use dozens. Nonetheless, the features used in the earliest systems continue to form the core of current SRL systems, and we begin by describing these core features as applied to FrameNet data by Gildea and Jurafsky (2002).

3.2.1 PHRASE TYPE

Different roles tend to be realized by different syntactic categories. For example, in FrameNet communication frames, the *Speaker* is likely to appear as a noun phrase, *Topic* as a prepositional phrase or noun phrase, and *Medium* as a prepositional phrase, as in: "[$_{\text{Speaker}}$We] talked [$_{\text{Topic}}$ about the proposal] [$_{\text{Medium}}$ over the phone] ."

The phrase type feature indicates the syntactic category of the phrase expressing the semantic roles, using the set of syntactic categories of the Penn Treebank project, as described in Marcus et al. (1993). In the FrameNet data, frame elements are most commonly expressed as noun phrases (NP, 47% of frame elements in the training set), and prepositional phrases (PP, 22%). The next most common categories are adverbial phrases (ADVP, 4%), particles (e.g., "make something *up*" — PRT, 2%) and clauses (SBAR, 2%, and S, 2%).

Gildea and Jurafsky (2002) used the parser of Collins (1997), a statistical parser trained on examples from the Penn Treebank, to generate parses of the same format for the sentences in the data. Phrase types were derived automatically from parse trees generated by the parser, as shown in

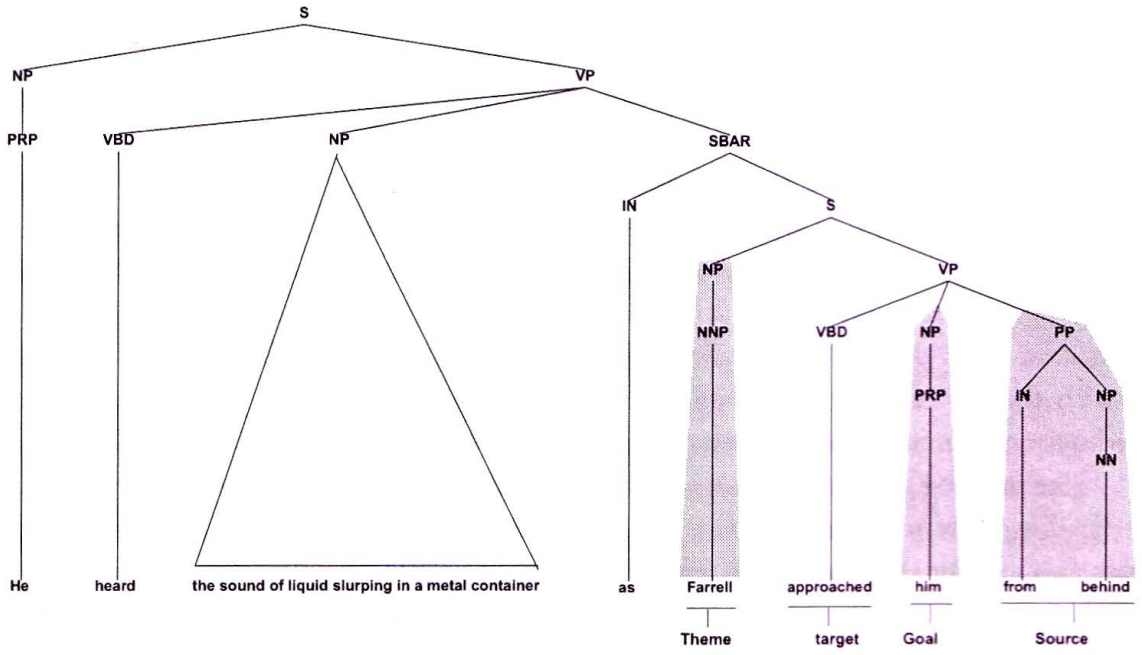


Figure 3.1: A sample sentence with parser output (above) and FrameNet annotation (below). Parse constituents corresponding to frame elements are highlighted.

Figure 3.1. Given the automatically generated parse tree, the constituent spanning the same set of words as each annotated frame element was found, and the constituent's nonterminal label was taken as the phrase type. In cases where more than one constituent matches due to a unary production in the parse tree, such as the NNP (proper noun) within the NP (noun phrase) over *Farrell* in Figure 3.1, the higher constituent was chosen.

The matching was performed by calculating the starting and ending word positions for each constituent in the parse tree, as well as for each annotated frame element, and matching each frame element with the parse constituent with the same beginning and ending points. Punctuation was ignored in this computation. Due to parsing errors, or, less frequently, mismatches between the parse tree formalism and the FrameNet annotation standards, there was sometimes no parse constituent matching an annotated frame element. In the FrameNet data, this occurred for 13% of the labels. The one case of systematic mismatch between the parse tree formalism and the FrameNet annotation standards is the FrameNet convention of including both a relative pronoun and its antecedent in frame elements, as in the first frame element in the following sentence:

(21) a. In its rough state he showed it to [$_{\text{Agt}}$ the Professor, who] **bent** [$_{\text{BPrt}}$ his grey beard] [$_{\text{Path}}$ over the neat script] and read for some time in silence.

Mismatch caused by the treatment of relative pronouns accounts for 1% of the roles in the FrameNet data.

During testing, the largest constituent beginning at the frame element's left boundary and lying entirely within the element was used to calculate the features. Gildea and Jurafsky (2002) did not use this technique on the training set, as it was expected to add noise to the data, but instead discarded examples with no matching parse constituent. The technique for finding a near match handles common parse errors such as a prepositional phrase being incorrectly attached to a noun phrase at the right-hand edge, and it guarantees that some syntactic category will be returned: the part-of-speech tag of the frame element's first word in the limiting case.

3.2.2 GOVERNING CATEGORY

The correlation between semantic roles and syntactic realization as subject or direct object is one of the primary facts that theories of Chapter 1 attempt to explain. As a basic example of how syntactic function is useful as a feature, in the sentence *He drove the car over the cliff*, the subject NP is more likely to fill the *Agent* role than the other two NPs. While some parsers produce trees annotated with such grammatical functions (Section 3.5.2), here we discuss grammatical function features that apply to syntactic trees in the standard Penn Treebank representation produced by parsers such as those of Collins (1997) and Charniak and Johnson (2005).

The first such feature, which we call "governing category," or *gov*, has only two values, S and VP, corresponding to subjects and objects of verbs, respectively. This feature is restricted to apply only to NPs as it was found to have little effect on other phrase types. As with phrase type, the feature was read from parse trees returned by the parser. We follow links from child to parent up the parse tree from the constituent corresponding to a frame element until either an S or VP node is found, and we assign the value of the feature according to whether this node is an S or VP. NP nodes found under S nodes are generally grammatical subjects, and NP nodes under VP nodes are generally objects. In most cases, the S or VP node determining the value of this feature immediately dominates the NP node, but attachment errors by the parser or constructions such as conjunction of two NPs can cause intermediate nodes to be introduced. Searching for higher ancestor nodes makes the feature robust to such cases. Even given good parses, this feature is not perfect in discriminating grammatical functions, and, in particular, it confuses direct objects with adjunct NP such as temporal phrases. For example, *town* in the sentence *He left town* and *yesterday* in the sentence *He left yesterday* will both be assigned a governing category of VP. Direct and indirect objects both appear directly under the VP node. For example, in the sentence *He gave me a new hose*, *me* and *a new hose* are both assigned a governing category of VP.

3.2.3 PARSE TREE PATH

Like the governing category feature described above, this feature is designed to capture the syntactic relation of a constituent to the rest of the sentence. However, the path feature describes the syntactic relation between the target word (that is, the predicate invoking the semantic frame) and the constituent in question, whereas the previous feature is independent of where the target word appears

in the sentence; that is, it identifies all subjects whether they are the subject of the target word or not.

This feature is defined as the **path** from the target word through the parse tree to the constituent in question, represented as a string of parse tree nonterminals linked by symbols indicating upward or downward movement through the tree, as shown in Figure 3.2. Although the path is composed of a string of symbols, the system will treat the string as an atomic value. The path includes, as the first element of the string, the part of speech of the target word, and, as the last element, the phrase type or syntactic category of the sentence constituent marked as a frame element. After some experimentation, Gildea and Jurafsky (2002) used a version of the path feature that collapses the various part-of-speech tags for verbs, including past tense verb (VBD), third person singular present verb (VBZ), other present tense verb (VBP), and past participle (VBN), into a single verb tag denoted "VB."

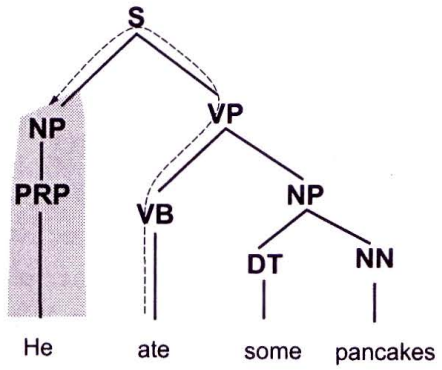


Figure 3.2: In this example, the **path** from the target word *ate* to the frame element *He* can be represented as VB↑VP↑S↓NP, with ↑ indicating upward movement in the parse tree and ↓ downward movement. The NP corresponding to *He* is found as described in Section 3.2.1.

The path feature is dependent on the syntactic representation used, which for most systems is the Treebank-2 annotation style (Marcus et al., 1994). Figure 3.3 shows the annotation for the sentence *They expect him to cut costs throughout the organization*, which exhibits the syntactic phenomenon known as subject-to-object raising, in which the main verb's object is interpreted as the embedded verb's subject. The Treebank-2 style tends to be generous in its usage of S nodes to indicate clauses, a decision intended to make possible a relatively straightforward mapping from S nodes to predications. In this example, the path from *cut* to the frame element *him* would be VB↑VP↑VP↑S↓NP, which typically indicates a verb's subject, despite the accusative case of the pronoun *him*. For the target word of *expect* in the sentence of Figure 3.3, the path to *him* would be VB↑VP↓S↓NP, rather than the typical direct object path of VB↑VP↓NP.

An example of Treebank-2 annotation of an "equi" construction, in which a noun phrase serves as an argument of both the main and subordinate verbs, is shown in Figure 3.4. Here, an empty category is used in the subject position of the subordinate clause, and is co-indexed with

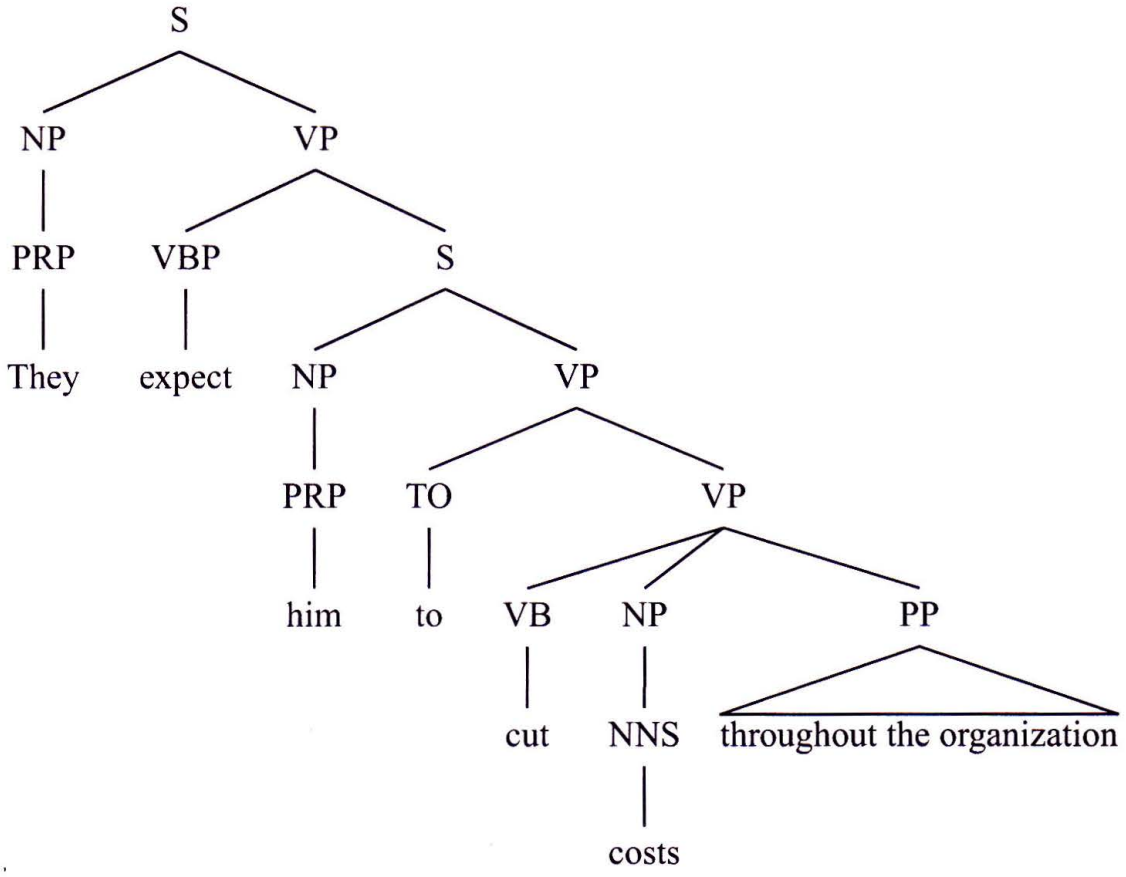


Figure 3.3: Treebank annotation of raising constructions.

the NP *Congress* in the direct object position of the main clause. The empty category, however, is not used in the statistical model of the parser or shown in its output. It is also not used by the FrameNet annotation, which would mark the NP *Congress* as a frame element of *raise* in this example. PropBank would mark both the empty category and the linked NP *Congress* with the role label. If we do not have the empty category, the value of the path feature from the target word *raise* to the frame element *Congress* would be VB↑VP↑VP↑S↑VP↓NP, and from the target word of *persuaded,* the path to *Congress* would be the standard direct object path VB↑VP↓NP.

The Treebank includes empty constituents for traces in various constructions, co-indexing relations between nodes, and secondary functional tags such as *subject* and *temporal*, all of which can help with interpretation of predicate-argument structure. However, most parser output does not include this additional information, but rather simply gives trees of phrase type categories. (For work on recovering this information automatically see Johnson (2002) and Dienes and Dubey (2003); some recent parsers have also included this information in their output (Gabbard et al., 2006).) The sentence of Figure 3.3 is one example of how the change in annotation style of Treebank-2 can affect this level of representation; the earlier style assigned the word *him* an NP node directly under the VP of *expect*.

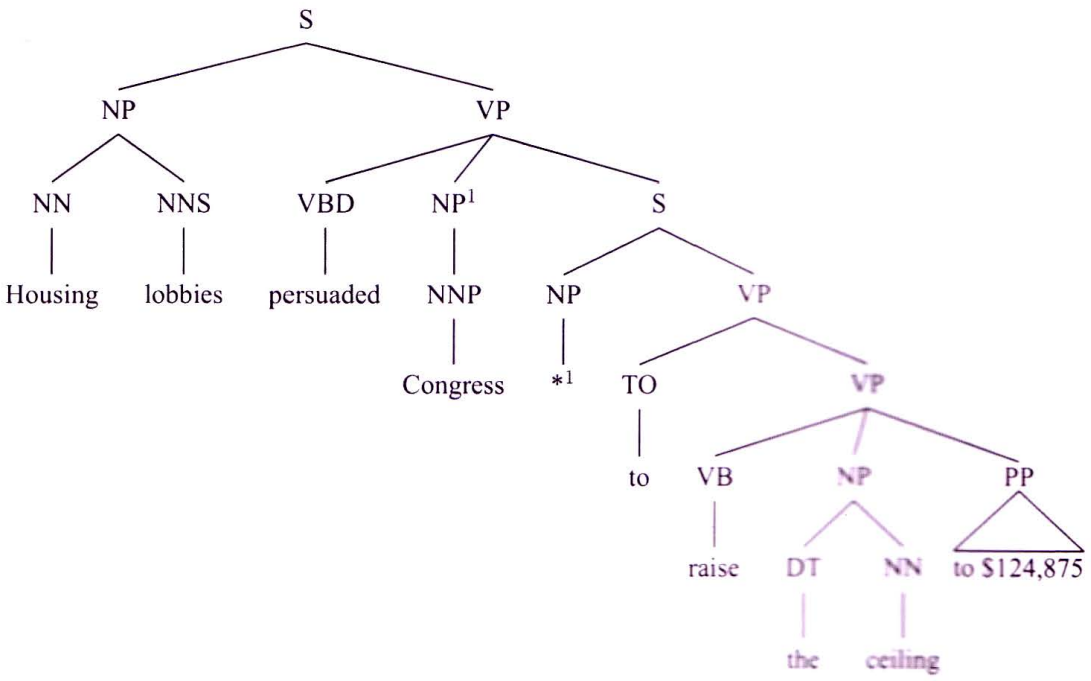


Figure 3.4: Treebank annotation of equi constructions. An empty category is indicated by *, and co-indexing by superscript [1].

The most common values of the path feature, along with interpretations, are shown in Table 3.1.

Table 3.1: Most frequent values of *path* feature in the training data.

Frequency	Path	Description
14.2%	VB↑VP↓PP	PP argument/adjunct
11.8	VB↑VP↑S↓NP	subject
10.1	VB↑VP↓NP	object
7.9	VB↑VP↑VP↑S↓NP	subject (embedded VP)
4.1	VB↑VP↓ADVP	adverbial adjunct
3.0	NN↑NP↑NP↓PP	prepositional complement of noun
1.7	VB↑VP↓PRT	adverbial particle
1.6	VB↑VP↑VP↑VP↑S↓NP	subject (embedded VP)
14.2		no matching parse constituent
31.4	Other	

For the purposes of choosing a frame element label for a constituent, the path feature is similar to the governing category feature defined above. Because the path captures more information, it may be more susceptible to parser errors and data sparseness. As an indication of this, the path feature

takes on a total of 2,978 possible values in the training data when not counting frame elements with no matching parse constituent, and 4,086 when finding paths to the best-matching constituent in these cases. The governing category feature, on the other hand, which is defined only for NPs, has only two values (S, corresponding to subjects, and VP, corresponding to objects). In cases in which the path feature includes an S or VP ancestor of an NP node as part of the path to the target word, the governing category feature is a function of the path feature. This is the case most of the time, including for the prototypical subject (VB↑VP↑S↓NP) and object (VB↑VP↓NP) paths. Of the 35,138 frame elements identified as NPs by the parser, only 4% have a path feature that does not include a VP or S ancestor. One such example is shown in Figure 3.5, where the small clause *the remainder renting...* has no S node, giving a path feature from *renting* to *the remainder* of *VB↑VP↑NP↓NP. The value of the governing category feature here is VP, as the algorithm finds the* VP of the sentence's main clause as it follows parent links up the tree, spuriously in this case, as the main VP is not headed by, or relevant to, the target word *renting*.

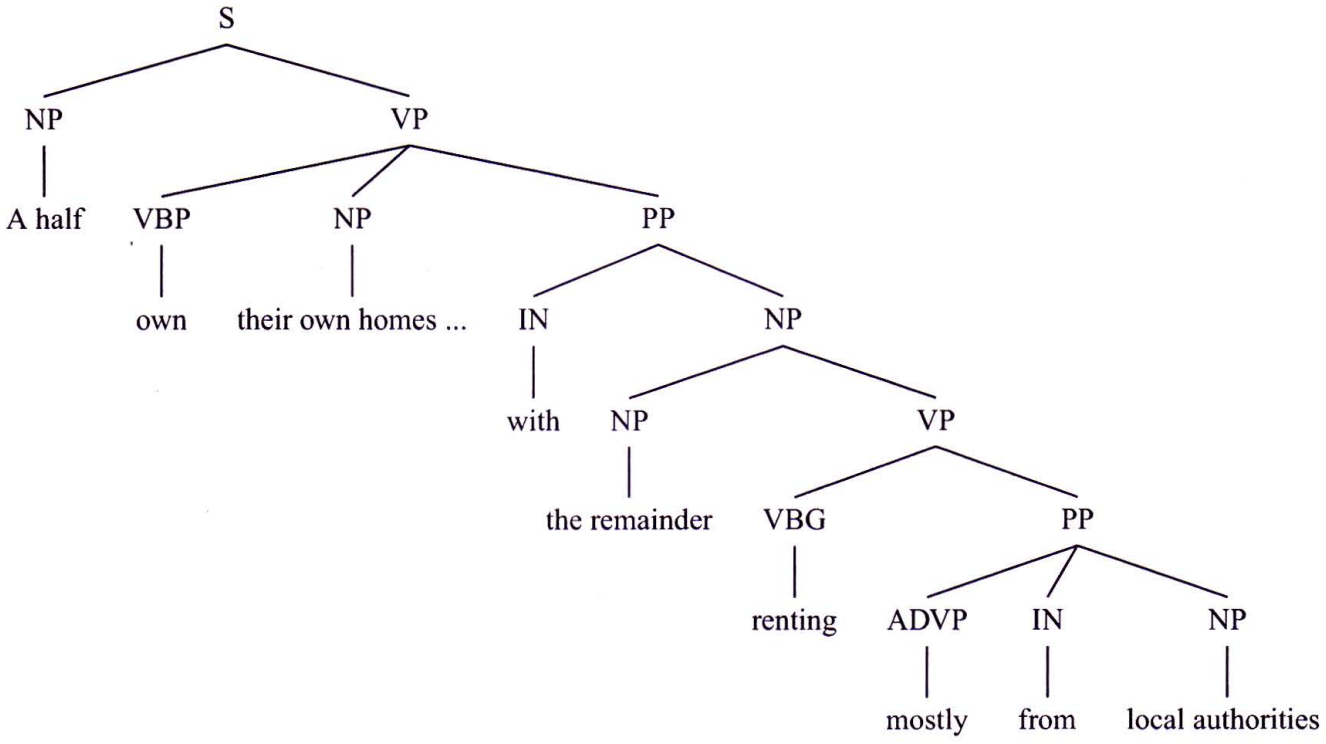


Figure 3.5: Example of target word *renting* in a small clause.

3.2.4 POSITION

In order to overcome errors due to incorrect parses, as well as to see how much can be done without parse trees, Gildea and Jurafsky (2002) use position as a feature. This feature simply indicates whether the constituent to be labeled occurs before or after the predicate defining the semantic frame. This

feature is highly correlated with grammatical function, since subjects will generally appear before a verb, and objects after.

Although we do not have hand-checked parses against which to measure the performance of the automatic parser on the FrameNet corpus, the result that 13% of frame elements have no matching parse constituent gives a rough idea of the parser's accuracy. Almost all of these cases are due to parser error. Other parser errors include cases in which a constituent is found, but with the incorrect label or internal structure. This measure also considers only the individual constituent representing the frame element — the parse for the rest of the sentence may be incorrect, resulting in an incorrect value for the grammatical function features described in the previous two sections. Collins (1997) reports 88% labeled precision and recall on individual parse constituents on data from the Penn Treebank, roughly consistent with the finding of at least 13% error.

3.2.5 VOICE

The distinction between active and passive verbs plays an important role in the connection between semantic role and grammatical function, since direct objects of active verbs often correspond in semantic role to subjects of passive verbs. From the parser output, verbs were classified as active or passive by building a set of 10 passive-identifying patterns. Each of the patterns requires both a passive auxiliary (some form of *to be* or *to get*) and a past participle. Roughly 5% of the FrameNet examples were identified as passive uses; Roland (2001) reports that 6.7% of verbs are passive in the Penn Treebank Wall Street Journal corpus, and 7.8% in the Brown corpus.

3.2.6 HEAD WORD

As previously noted, lexical dependencies are extremely important in labeling semantic roles as indicated by their importance in related tasks such as parsing. Head words of noun phrases can be used to express selectional restrictions on the semantic types of role fillers. For example, in a communication frame, noun phrases headed by *Bill*, *brother*, or *he* are more likely to be the *Speaker*, while those headed by *proposal*, *story*, or *question* are more likely to be the *Topic*. (Most systems do not attempt to resolve pronoun references.)

Since the parser assigns each constituent a head word as an integral part of the parsing model, we can read the head words of the constituents from the parser output, using the same set of rules for identifying the head child of each constituent in the parse tree. The head word rules are listed in Collins (1999). Prepositions are considered to be the head words of prepositional phrases. The head word rules do not attempt to distinguish between cases in which the preposition expresses the semantic content of a role filler, such as PATH frame elements expressed by prepositional phrases headed by *along*, *through*, or *in*, and cases in which the preposition might be considered to be purely a case marker, as in most uses of *of*, where the semantic content of the role filler is expressed by the preposition's object. Complementizers are considered to be heads, meaning that infinitive verb phrases are always headed by *to*, and subordinate clauses such as in the sentence *I'm sure that he came* are headed by *that*.

3.2.7 SUBCATEGORIZATION

This feature refers to the set of a verb's syntactic arguments in the sentence. For example, we expect an intransitive use of *close* such as the *The door closed* to have a different mapping from syntactic to semantic roles when compared to the transitive use in *He closed the door*. The subcategorization feature of the first verb usage is { subject }, while the subcategorization for the second is { subject, object }.

3.2.8 ARGUMENT SET

The feature, called Frame Element Group in the FrameNet-based system of Gildea and Jurafsky (2002), is the set of all roles appearing for a verb in a given sentence. Since this feature depends on the roles assigned to all constituents in a sentence, it was employed in a post-processing ranking of role assignments for an entire sentence.

3.2.9 FEATURES INTRODUCED IN LATER SYSTEMS

We now turn from the features of Gildea and Jurafsky (2002) to those introduced by later systems; while we cannot describe every SRL system in the literature, we will try to list the features which have been found to be most important in SRL systems. The majority of these systems are trained and tested on the PropBank data; we discuss some of the issues related to the differences between the FrameNet and PropBank annotation schemes in Section 3.8.

Argument Order This feature, introduced by Fleischman et al. (2003), is an integer indicating the position of a constituent in the sequence of arguments for the given verb. It is computed after an initial phase classifies constituents as arguments or non-arguments. Because the feature does not use the syntactic parse tree, it can help make a semantic role labeling system robust to parser error.

Previous Role This feature, introduced by Fleischman et al. (2003), is simply the label assigned by the system to the previous argument (skipping non-argument constituents). Because this feature introduces a dependency among labels assigned to different constituents, it requires an HMM-style Viterbi search (Section 3.4.2) to find the best overall sequence for all labels in a sentence.

Head Word Part of Speech This feature was found to boost performance by Surdeanu et al. (2003). Because Penn Treebank part of speech categories distinguish singular from plural nouns, and proper and common nouns, the part of speech tags of the head of an NP can refine the type of noun phrase involved.

Named entities in Constituents The first of a number of features introduced by Pradhan et al. (2005), this feature uses the named entity recognizer of Bikel et al. (1999) to identify words as

instances of the classes PERSON, ORGANIZATION, LOCATION, PERCENT, MONEY, TIME, and DATE. This feature helps handle the data sparsity caused by the unlimited sets of proper names for people, organizations, and locations in particular.

Verb Clustering An automatic clustering of verbs is derived by clustering verbs according to the direct objects with which they appear using Expectation-Maximization over a latent variable model (Hofmann and Puzicha, 1998; Rooth et al., 1999). Because semantically similar verbs such as *eat* and *devour* will occur with the same objects, they will be assigned to the same clusters. The use of this cluster for predicting argument structure is motivated by the observation that semantically similar verbs undergo the same pattern of argument alternation (Levin, 1993), as discussed in Section 1.3.3.

Head Word of Objects of PPs When a verb's argument is a prepositional phrase (PP), the PP's head word is the preposition. While this can often be a reliable indicator of semantic role (for example *in*, *across*, and *toward* usually indicate location), most prepositions can be used in many different ways, and the sense can be determined by the preposition's object. For example, *in February* indicates time, while *in New York* indicates location.

First/Last word/POS in Constituent As with the previous feature, this feature provides more specific information about the argument filler than the headword alone, but does so in a more general way that is robust to parser error and applies to any type of constituent.

Constituent Order This feature is related to the argument order feature above but is designed to be used in discriminating arguments from non-arguments. Pradhan et al. (2005) use a version of this feature where the position of each constituent is calculated relative to the predicate, which helps favor constituents close to the predicate.

Constituent Tree Distance This feature has the same motivation as the previous feature but takes syntactic structure into account.

Constituent Context Features These features provide information about the parent and left and right siblings of a constituent. For each of these three constituents, the phrase type, head word, and the head word's part of speech are given as features, for a total of nine features.

Temporal Cue Words A number of words which indicate time, but which are not considered named entities by the named entity tagger, are included as binary features.