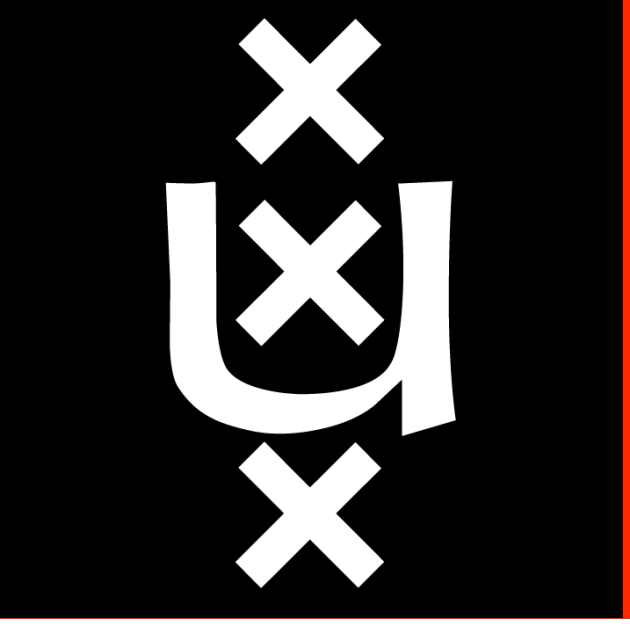


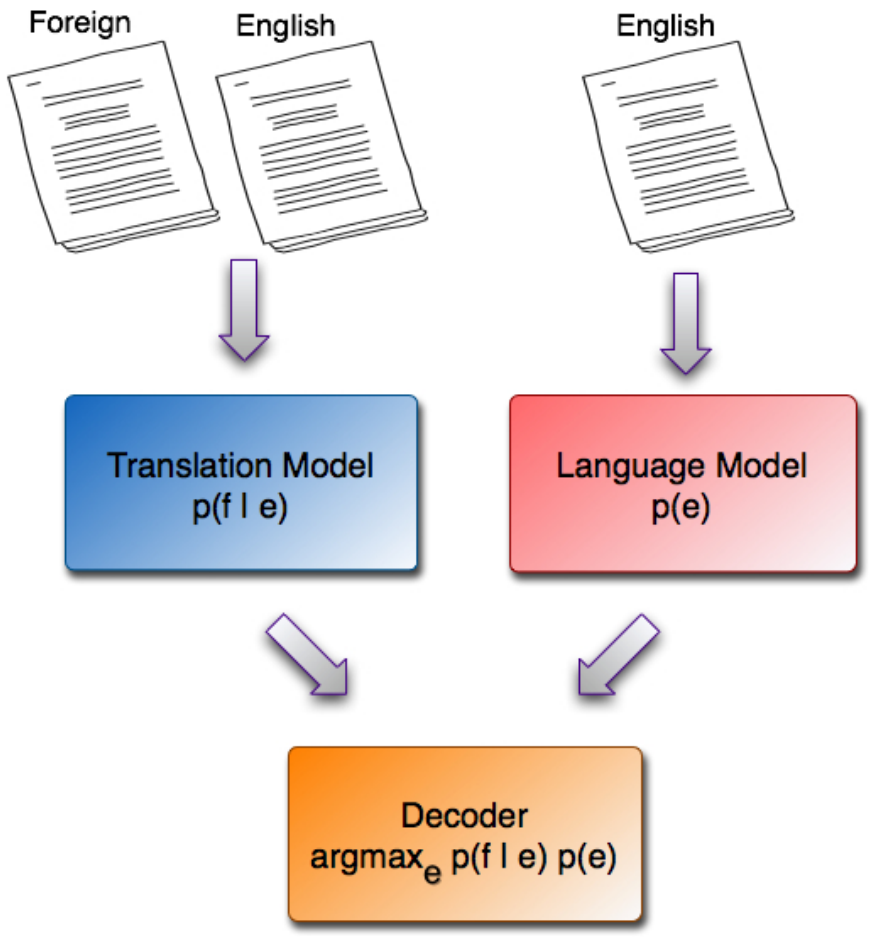
Christof Monz

Applied Language Technology
Decoding

Today's Class

- Decoding as search
- Decoding complexity
- Search space restrictions and pruning
- N-best lists and Minimum Bayes Risk
- Search errors vs. model errors
- Cube Pruning
- System combination

General SMT Architecture



Log-linear model: $p(e|f) = \exp(\sum_{m=1}^{M} \lambda_m h_m(e,f))$

What's Decoding?

- $\text{trans}(f) = \arg\max_e p(f|e) \cdot p(e)$
  - Conveniently, we did not specify the domain of the $\arg\max$ operator
- Assume that the vocabulary (V) of the target language (English) is known, e can be drawn from the set of all possible sequences E^* consisting of words from V
  - $|E^*| = \sum_{l=1}^{\infty} |V|^l$
- Clearly E^* is way too under-specified
  - *(a)* Most sequences in E^* are not syntactically well-formed $\rightarrow p(e) \approx 0$
  - *(b)* Most sequences in E^* are not semantically equivalent to f: $\rightarrow p(f|e) \approx 0$
  - *(c)* There are no sentences of lengths approaching ∞
- Restrict to $\tilde{E} \subset E^*$ taking (a–c) into account

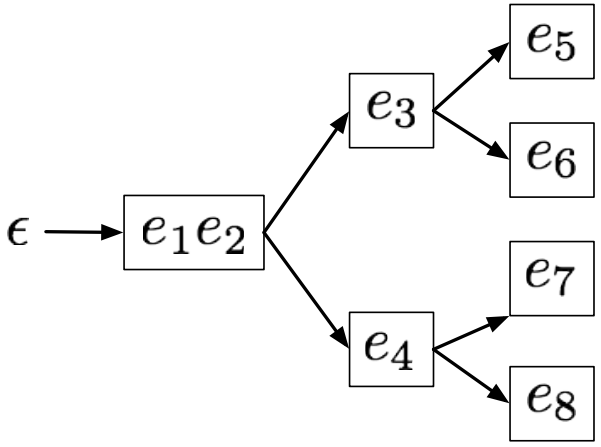
Decoding as Search

- How can we define $\tilde{E} \subset E^*$?
- Instead of assuming a fixed set $\tilde{E}$, we incrementally generate prefix sequences $\tilde{e}$ of increasing lengths s.t.
  - for some possible continuation e^* of $\tilde{e}$: $\tilde{e} \oplus e^* \in \tilde{E}$

    where $\tilde{e} \oplus e^*$ is the concatenation of $\tilde{e}$ and e^*
  - $\text{trans}(f) = \arg\max_{e \in \tilde{E}} p(f|e) \cdot p(e)$
- In practice, this is cast as a search problem, where we search within E^* for the prefix sequences that
  - can be be *generated* by the translation, language, and reordering models
  - *score* highly, i.e., are likely, given the model parameters

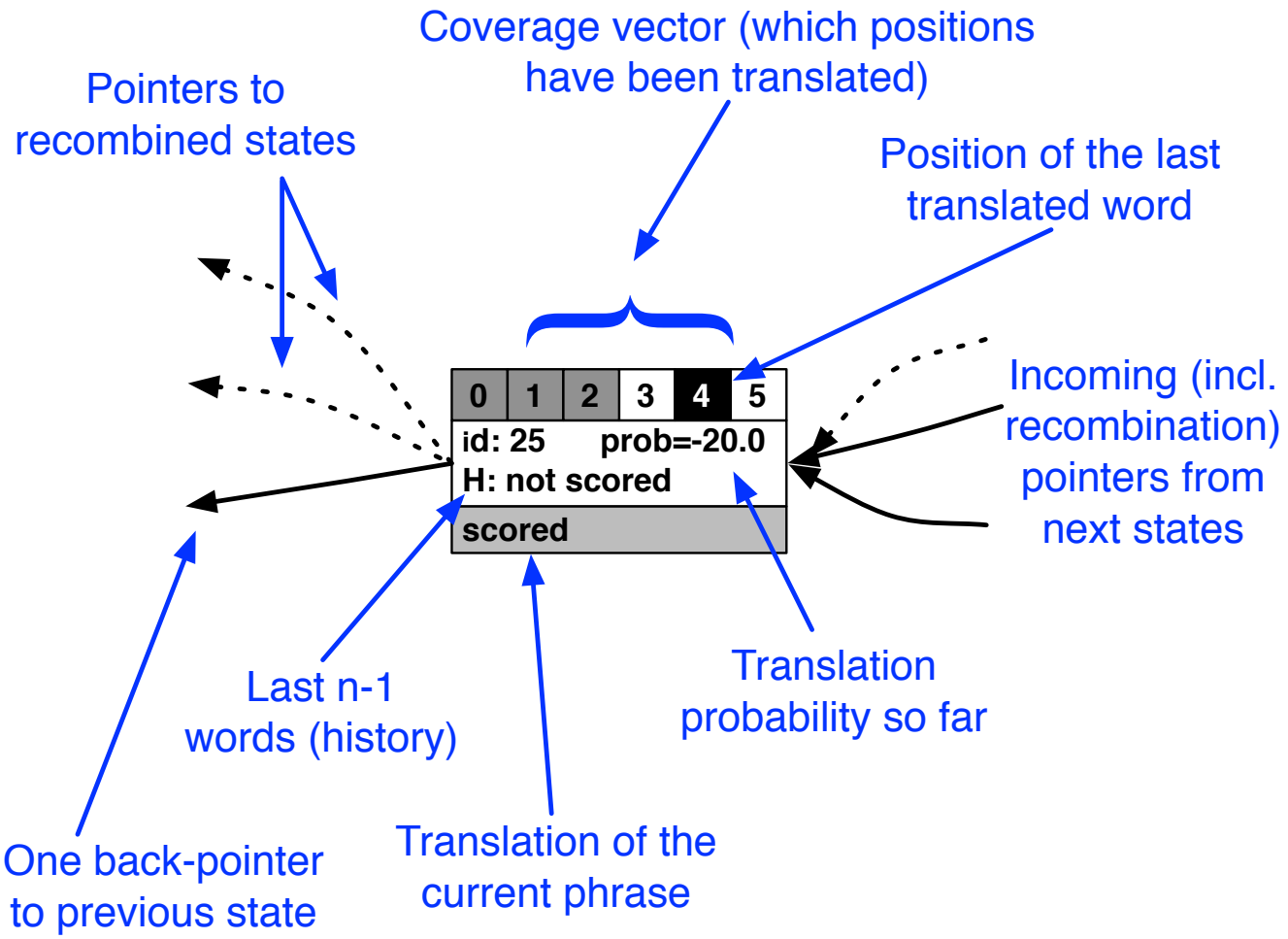
Search Space Representation

- Assume that $\tilde{E} = \left\{ \begin{array}{l} e_1 e_2 e_3 e_5, \\ e_1 e_2 e_3 e_6, \\ e_1 e_2 e_4 e_7, \\ e_1 e_2 e_4 e_8 \end{array} \right\}$
  - when scoring the different sequences, we repeatedly score the common prefixes $e_1 e_2$ (4x), $e_1 e_2 e_3$ (2x), and $e_1 e_2 e_4$ (2x)
- Instead the we can represent this set more compactly as a prefix tree where each node in the tree corresponds to a *state* and branches to state transitions



Decoding States

Peter	*has*	*not-a*	*goal*	*scored*	*.*
0	1	2	3	4	5
Peter	**hat**	**kein**	**Tor**	**geschossen**	**.**

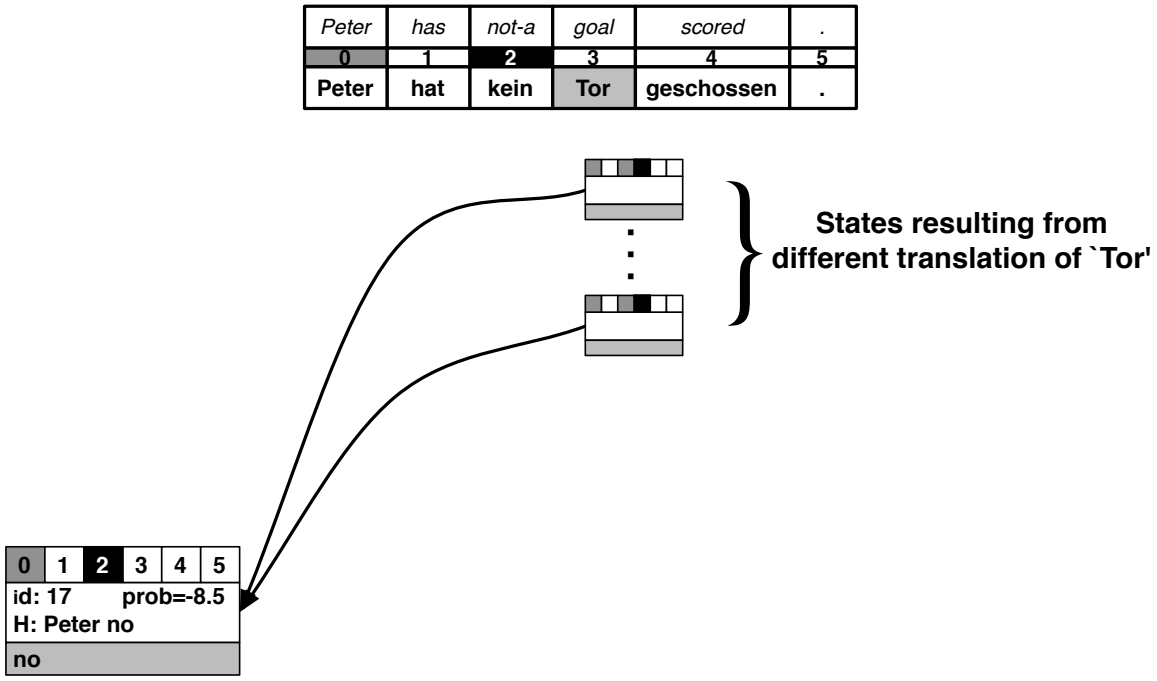


State Transitions

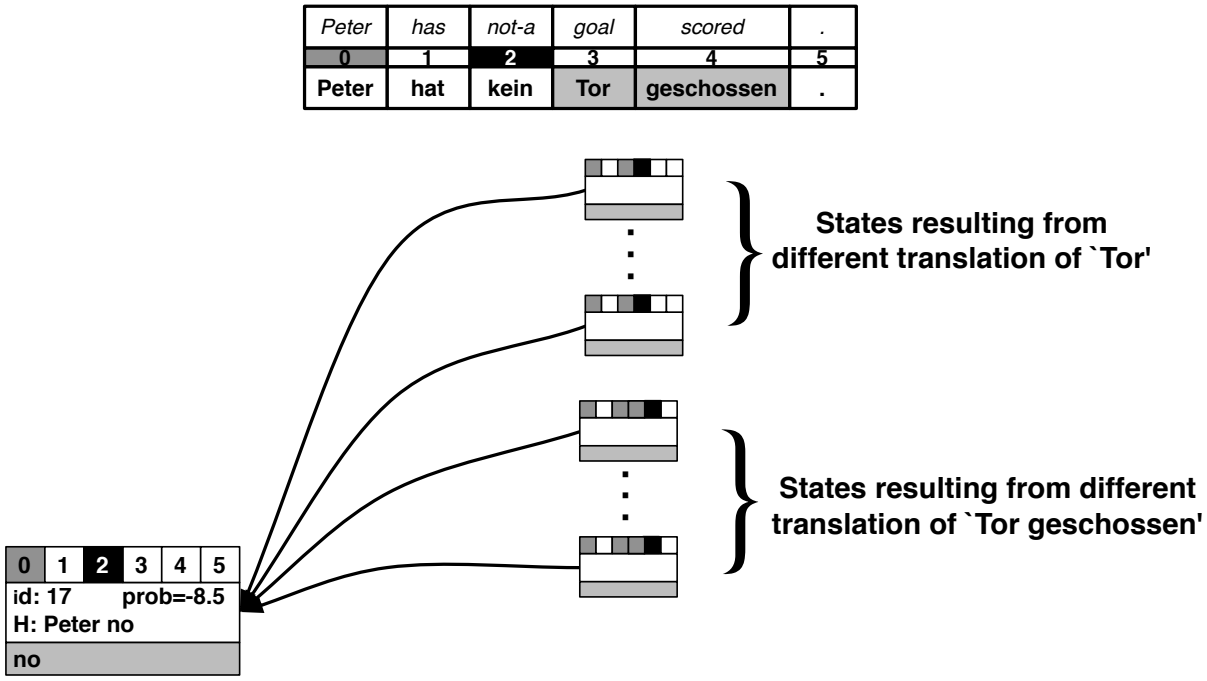
Peter	*has*	*not-a*	*goal*	*scored*	*.*
0	1	2	3	4	5
Peter	**hat**	**kein**	**Tor**	**geschossen**	**.**

0	1	2	3	4	5
id: 17 prob=-8.5 **H: Peter no**					
no					

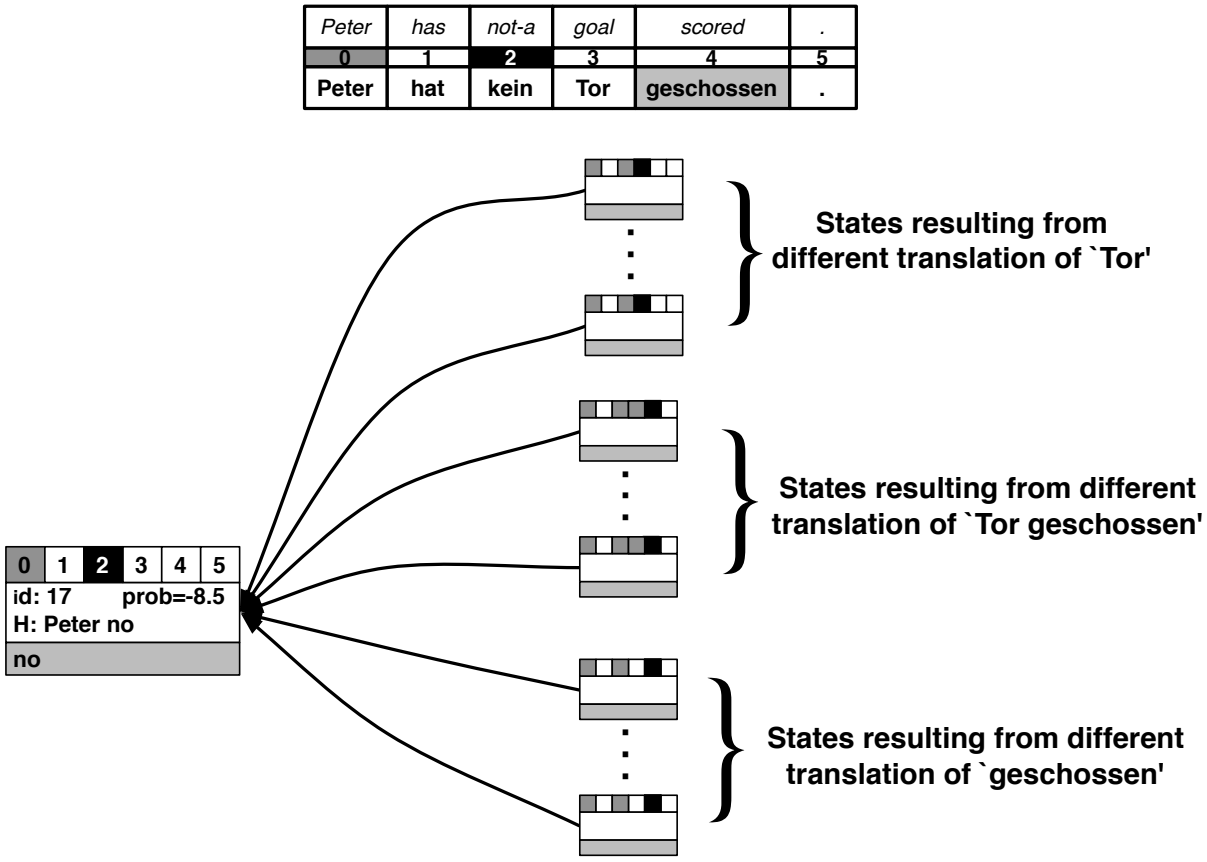
State Transitions



State Transitions

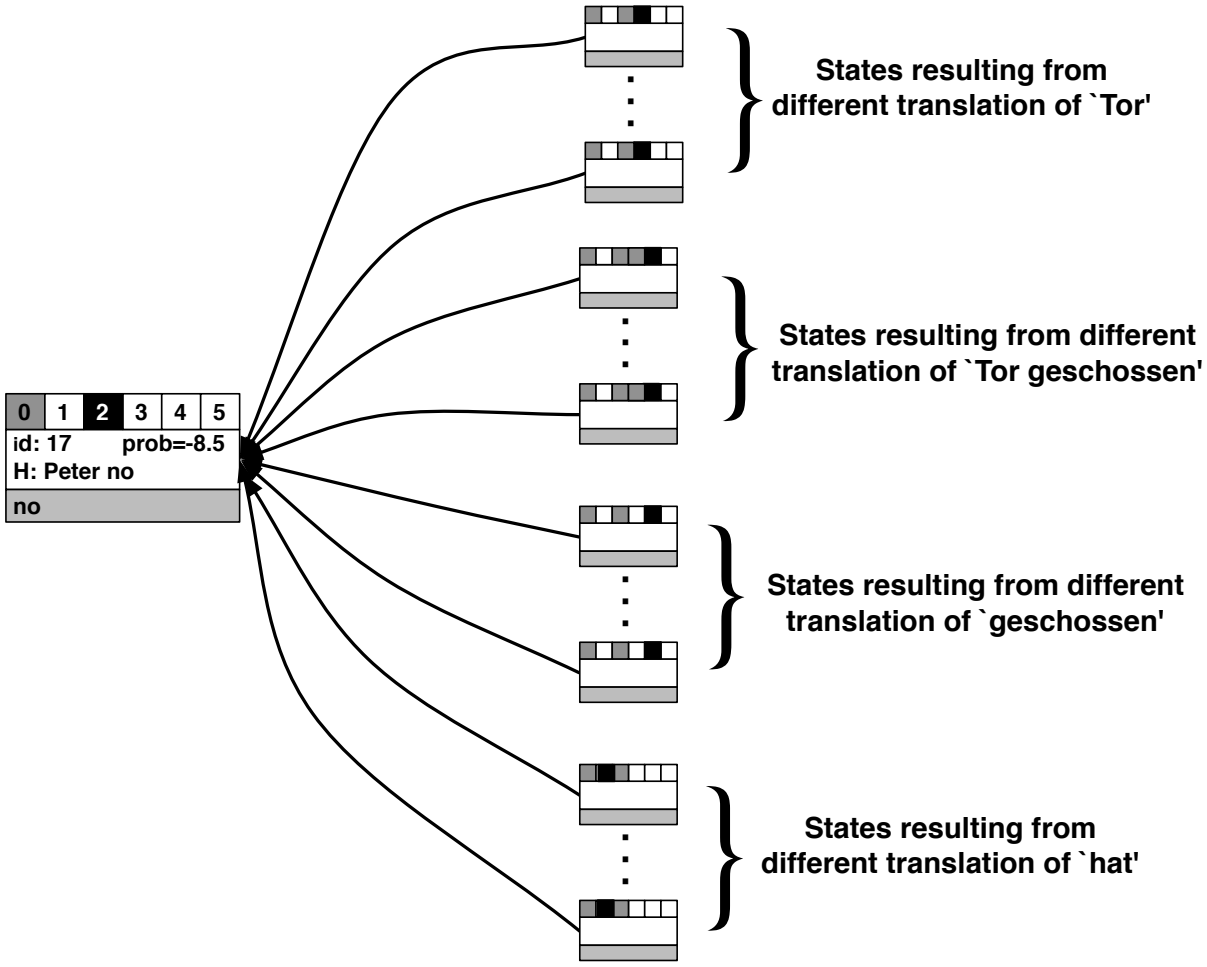


State Transitions



State Transitions

Peter	has	not-a	goal	scored	.
0	1	2	3	4	5
Peter	hat	kein	Tor	geschossen	.



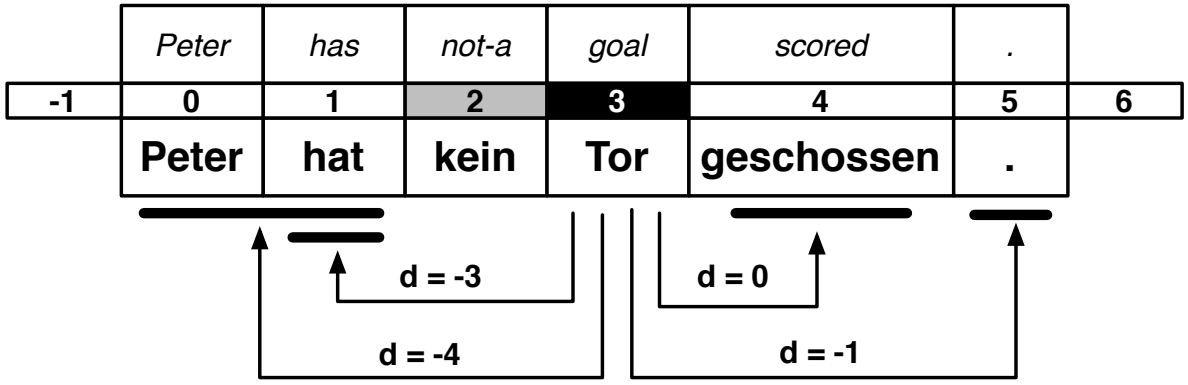
Transition Cost Functions

- Whenever a state s_i expands to a new state s_j we compute the following transition costs:
- **Phrase Translation:** $h_{TM}(s_j) = p(\bar{e}_j|\bar{f}_j) + p(\bar{f}_j|\bar{e}_j)$

  where $\bar{f}_j$ is the foreign phrase covered by s_j and e_j is the corresponding translation according to the translation model
- **LM Continuation:** $h_{LM}(s_i, s_j) = \sum_{k=1}^{\text{length}(\bar{e}_j)} p(w_k|c(k, s_i, \bar{e}_j))$
  - $c(k, s_i, \bar{e}_j)$ generates the appropriate sliding window history
- **Phrase Penalty:** Each phrase application incurs a fixed cost, typically $h_{PP}(s_j) = -1$
- **Word Penalty:** The LM prefers shorter translations. To compensate for that we define $h_{WP}(s_j) = \text{length}(\bar{e}_j)$. Note, this is a positive value (and "word bonus" would have been more appropriate).

Transition Cost Functions

- **Linear Distortion:** This cost reflects the degree to which state transitions deviate from a strict monotonic left-to-right translation
  - $h_{LD}(s_i, s_j) = -1 \cdot |\text{first\_pos}(s_j) - \text{last\_pos}(s_i) - 1|$



- There a many more advanced reordering models
  - We come back to this later
  - Almost all of them are used in addition to linear distortion

Decoding Example

- Let's take a look at an example!
- Disclaimer!
  - The following example is intended purely to illustrate the mechanics of the search process during decoding
  - The translation and language model probabilities are made up, are incomplete, and have no empirical grounding
    - For instance, some n-gram counts of order $n-1$ are ignored, while we use counts of order n
    - I did not check whether conditional probabilities add up to 1
  - Some costs such as *word penalty* and direct translation probabilities have been ignored
- Nevertheless, this does not affect the principles of the underlying search procedure
- For the following example we assume a beam width of -2.9
- We also assume $\lambda_m = 1$ for $1 \leq m \leq M$

Decoding Example

Translation Model

```
p( Peter| Peter ) = -1.5
p( has  | hat   ) = -1
p( did  | hat   ) = -2
p( no   | kein  ) = -1
p( did not |hat kein)=-0.5
p( has no  |hat kein ) = -1.5
p( goal| Tor) = -2
p( goals|Tor ) = -3
p( shot  |geschossen ) = -1
p( scored|geschossen ) = -2
p( shoot |geschossen ) = -1
p(scored .|Tor geschossen .)=-10
p(.|.) = -0.5
```

	Peter	*has*	*not-a*	*goal*	*scored*	*.*	
-1	0	1	2	3	4	5	6
	Peter	**hat**	**kein**	**Tor**	**geschossen**	**.**	

Language Model

ARPA format

```
-0.5  .                -3
-4    goal             -0.5
-2    not              -1
-1    Peter
-6    scored           -3
-5    shoot            -0.1
-0.5  . </s>
-2    <s> Peter        -1
-3    <s> has          -0.5
-1    <s> did          -0.7
-0.5  did not          -0.5
-1.5  has no           -2
-2    shoot goals      -1
-1    <s> Peter did
-1    not shoot goals
-0.5  Peter did not
```

0	1	2	3	4	5
id: 0					prob=0
H: <s>					

Decoding Example

Translation Model

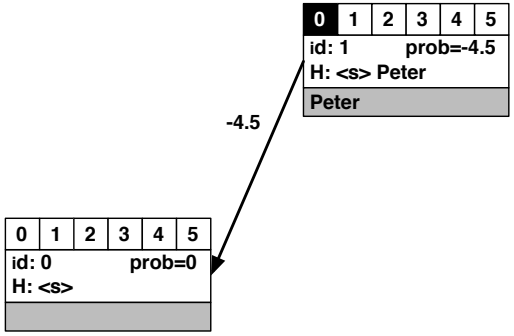
```
p( Peter| Peter ) = -1.5
p( has  | hat   ) = -1
p( did  | hat   ) = -2
p( no   | kein  ) = -1
p( did not |hat kein)=-0.5
p( has no  |hat kein ) = -1.5
p( goal| Tor) = -2
p( goals|Tor ) = -3
p( shot  |geschossen ) = -1
p( scored|geschossen ) = -2
p( shoot |geschossen ) = -1
p(scored .|Tor geschossen .)=-10
p(.|.) = -0.5
```

	Peter	*has*	*not-a*	*goal*	*scored*	*.*	
-1	0	1	2	3	4	5	6
	Peter	**hat**	**kein**	**Tor**	**geschossen**	**.**	

Language Model

ARPA format

```
-0.5  .                -3
-4    goal             -0.5
-2    not              -1
-1    Peter
-6    scored           -3
-5    shoot            -0.1
-0.5  . </s>
-2    <s> Peter        -1
-3    <s> has          -0.5
-1    <s> did          -0.7
-0.5  did not          -0.5
-1.5  has no           -2
-2    shoot goals      -1
-1    <s> Peter did
-1    not shoot goals
-0.5  Peter did not
```



Decoding Example

Translation Model

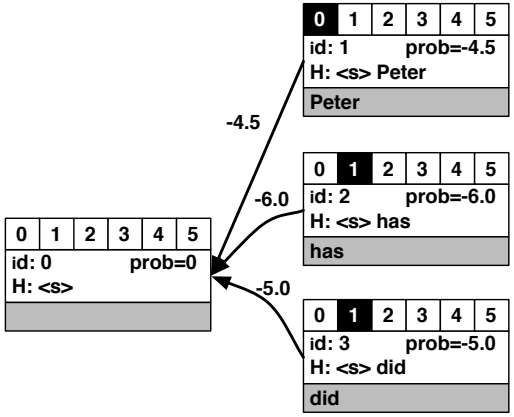
```
p( Peter| Peter ) = -1.5
p( has  | hat   ) = -1
p( did  | hat   ) = -2
p( no   | kein  ) = -1
p( did not |hat kein)=-0.5
p( has no  |hat kein ) = -1.5
p( goal| Tor) = -2
p( goals|Tor ) = -3
p( shot  |geschossen ) = -1
p( scored|geschossen ) = -2
p( shoot |geschossen ) = -1
p(scored .|Tor geschossen .)=-10
p(.|.) = -0.5
```

	Peter	*has*	*not-a*	*goal*	*scored*	*.*	
-1	0	1	2	3	4	5	6
	Peter	**hat**	**kein**	**Tor**	**geschossen**	**.**	

Language Model

ARPA format

```
-0.5  .                -3
-4    goal             -0.5
-2    not              -1
-1    Peter
-6    scored           -3
-5    shoot            -0.1
-0.5  . </s>
-2    <s> Peter        -1
-3    <s> has          -0.5
-1    <s> did          -0.7
-0.5  did not          -0.5
-1.5  has no           -2
-2    shoot goals      -1
-1    <s> Peter did
-1    not shoot goals
-0.5  Peter did not
```



Decoding Example

Translation Model

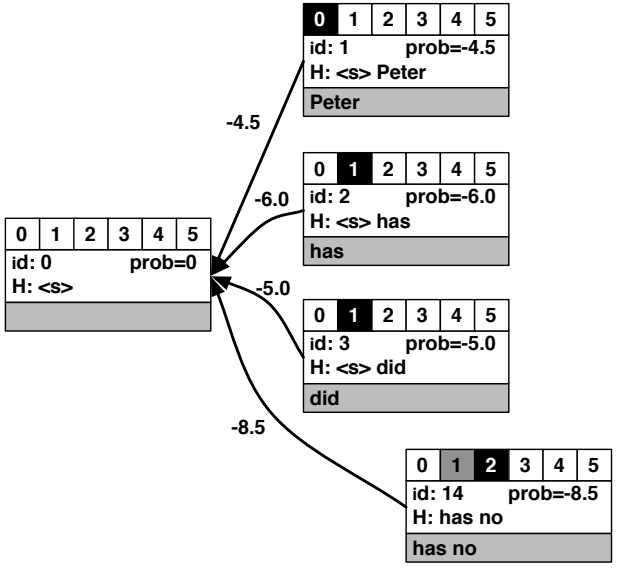
```
p( Peter| Peter ) = -1.5
p( has  | hat   ) = -1
p( did  | hat   ) = -2
p( no   | kein  ) = -1
p( did not |hat kein)=-0.5
p( has no  |hat kein ) = -1.5
p( goal| Tor) = -2
p( goals|Tor ) = -3
p( shot  |geschossen ) = -1
p( scored|geschossen ) = -2
p( shoot |geschossen ) = -1
p(scored .|Tor geschossen .)=-10
p(.|.) = -0.5
```

	Peter	*has*	*not-a*	*goal*	*scored*	*.*	
-1	0	1	2	3	4	5	6
	Peter	**hat**	**kein**	**Tor**	**geschossen**	**.**	

Language Model

ARPA format

```
-0.5  .                -3
-4    goal             -0.5
-2    not              -1
-1    Peter
-6    scored           -3
-5    shoot            -0.1
-0.5  . </s>
-2    <s> Peter        -1
-3    <s> has          -0.5
-1    <s> did          -0.7
-0.5  did not          -0.5
-1.5  has no           -2
-2    shoot goals      -1
-1    <s> Peter did
-1    not shoot goals
-0.5  Peter did not
```



Decoding Example

Translation Model

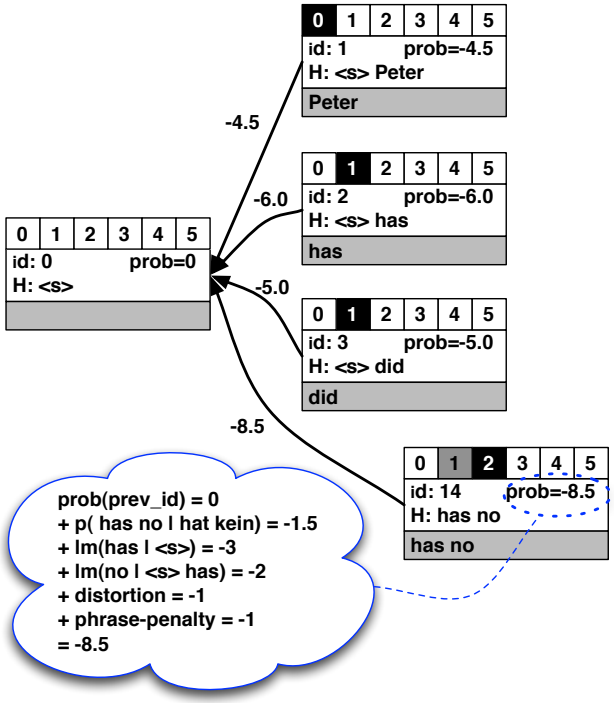
```
p( Peter| Peter ) = -1.5
p( has  | hat   ) = -1
p( did  | hat   ) = -2
p( no   | kein  ) = -1
p( did not |hat kein)=-0.5
p( has no  |hat kein ) = -1.5
p( goal| Tor) = -2
p( goals|Tor ) = -3
p( shot  |geschossen ) = -1
p( scored|geschossen ) = -2
p( shoot |geschossen ) = -1
p(scored .|Tor geschossen .)=-10
p(.|.) = -0.5
```

	Peter	*has*	*not-a*	*goal*	*scored*	*.*	
-1	0	1	2	3	4	5	6
	Peter	**hat**	**kein**	**Tor**	**geschossen**	**.**	

Language Model

ARPA format

```
-0.5  .                -3
-4    goal             -0.5
-2    not              -1
-1    Peter
-6    scored           -3
-5    shoot            -0.1
-0.5  . </s>
-2    <s> Peter        -1
-3    <s> has          -0.5
-1    <s> did          -0.7
-0.5  did not          -0.5
-1.5  has no           -2
-2    shoot goals      -1
-1    <s> Peter did
-1    not shoot goals
-0.5  Peter did not
```



Decoding Example

Translation Model

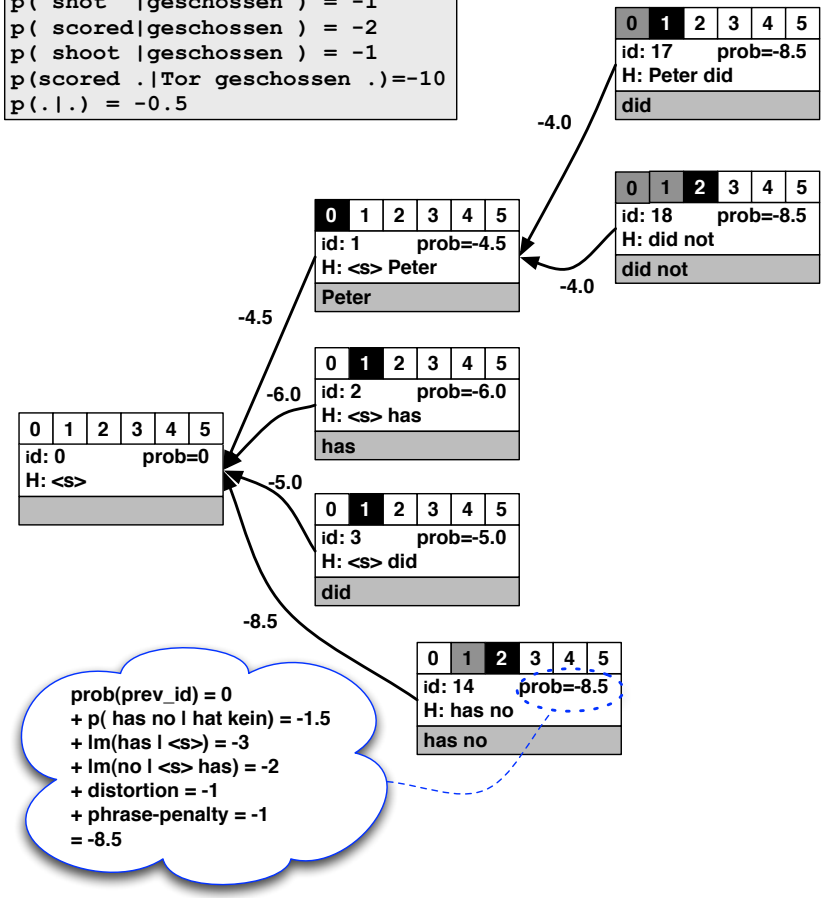
```
p( Peter| Peter ) = -1.5
p( has  | hat   ) = -1
p( did  | hat   ) = -2
p( no   | kein  ) = -1
p( did not |hat kein)=-0.5
p( has no  |hat kein ) = -1.5
p( goal| Tor) = -2
p( goals|Tor ) = -3
p( shot  |geschossen ) = -1
p( scored|geschossen ) = -2
p( shoot |geschossen ) = -1
p(scored .|Tor geschossen .)=-10
p(.|.) = -0.5
```

	Peter	*has*	*not-a*	*goal*	*scored*	*.*	
-1	0	1	2	3	4	5	6
	Peter	**hat**	**kein**	**Tor**	**geschossen**	**.**	

Language Model

ARPA format

```
-0.5  .                -3
-4    goal             -0.5
-2    not              -1
-1    Peter
-6    scored           -3
-5    shoot            -0.1
-0.5  . </s>
-2    <s> Peter        -1
-3    <s> has          -0.5
-1    <s> did          -0.7
-0.5  did not          -0.5
-1.5  has no           -2
-2    shoot goals      -1
-1    <s> Peter did
-1    not shoot goals
-0.5  Peter did not
```



Decoding Example

Translation Model

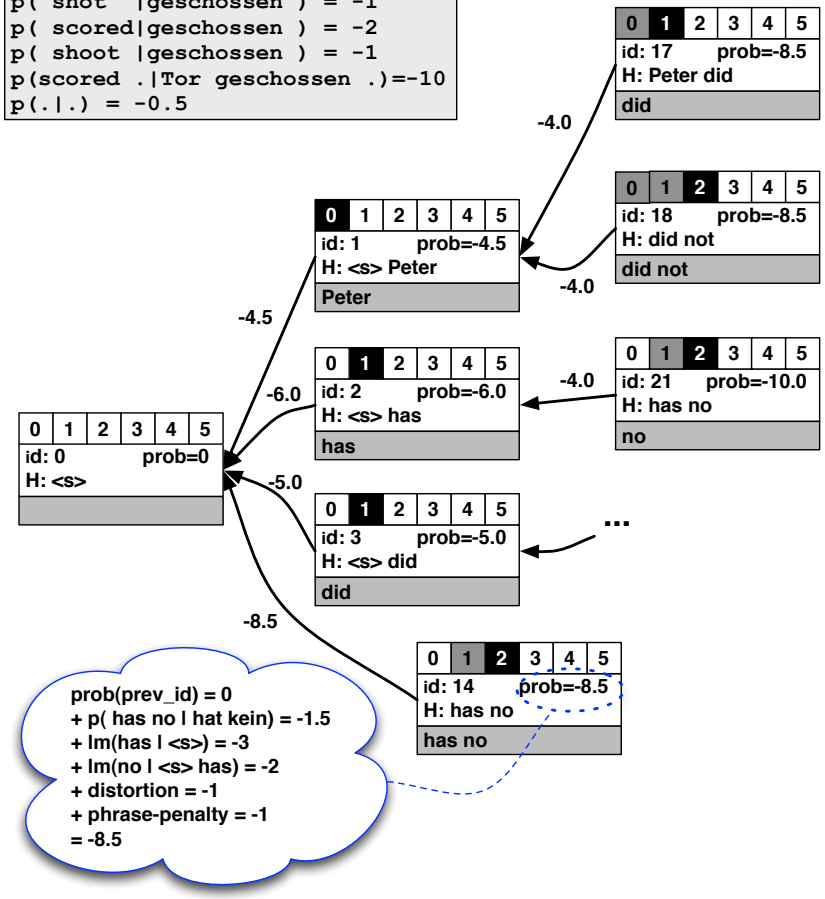
```
p( Peter| Peter ) = -1.5
p( has  | hat   ) = -1
p( did  | hat   ) = -2
p( no   | kein  ) = -1
p( did not |hat kein)=-0.5
p( has no  |hat kein ) = -1.5
p( goal| Tor) = -2
p( goals|Tor ) = -3
p( shot  |geschossen ) = -1
p( scored|geschossen ) = -2
p( shoot |geschossen ) = -1
p(scored .|Tor geschossen .)=-10
p(.|.) = -0.5
```

	Peter	*has*	*not-a*	*goal*	*scored*	*.*	
-1	0	1	2	3	4	5	6
	Peter	**hat**	**kein**	**Tor**	**geschossen**	**.**	

Language Model

ARPA format

```
-0.5  .                -3
-4    goal             -0.5
-2    not              -1
-1    Peter
-6    scored           -3
-5    shoot            -0.1
-0.5  . </s>
-2    <s> Peter        -1
-3    <s> has          -0.5
-1    <s> did          -0.7
-0.5  did not          -0.5
-1.5  has no           -2
-2    shoot goals      -1
-1    <s> Peter did
-1    not shoot goals
-0.5  Peter did not
```



Decoding Example

Translation Model

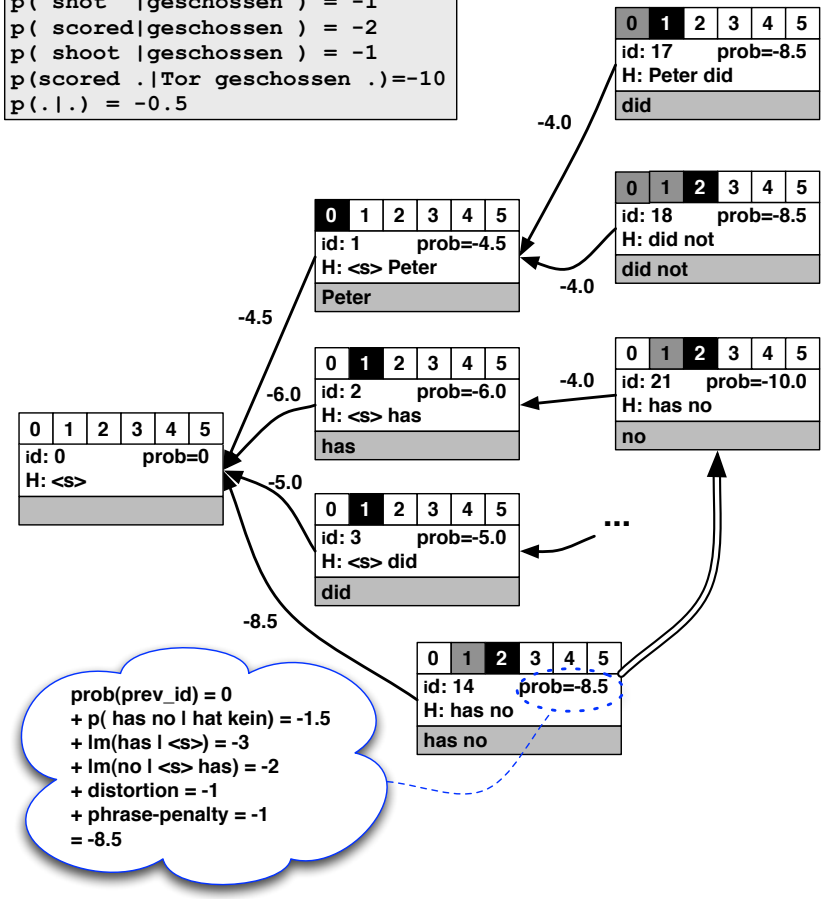
```
p( Peter| Peter ) = -1.5
p( has  | hat   ) = -1
p( did  | hat   ) = -2
p( no   | kein  ) = -1
p( did not |hat kein)=-0.5
p( has no  |hat kein ) = -1.5
p( goal| Tor) = -2
p( goals|Tor ) = -3
p( shot  |geschossen ) = -1
p( scored|geschossen ) = -2
p( shoot |geschossen ) = -1
p(scored .|Tor geschossen .)=-10
p(.|.) = -0.5
```

	Peter	*has*	*not-a*	*goal*	*scored*	*.*	
-1	0	1	2	3	4	5	6
	Peter	**hat**	**kein**	**Tor**	**geschossen**	**.**	

Language Model

ARPA format

```
-0.5  .                -3
-4    goal             -0.5
-2    not              -1
-1    Peter
-6    scored           -3
-5    shoot            -0.1
-0.5  . </s>
-2    <s> Peter        -1
-3    <s> has          -0.5
-1    <s> did          -0.7
-0.5  did not          -0.5
-1.5  has no           -2
-2    shoot goals      -1
-1    <s> Peter did
-1    not shoot goals
-0.5  Peter did not
```



Decoding Example

Translation Model

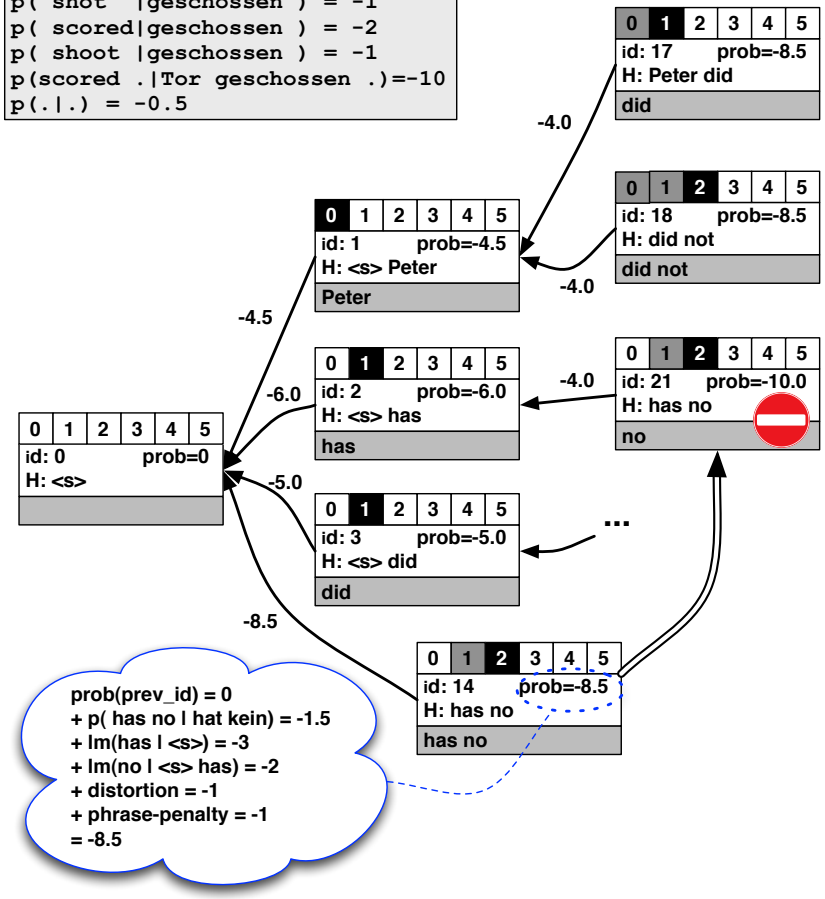
```
p( Peter| Peter ) = -1.5
p( has  | hat   ) = -1
p( did  | hat   ) = -2
p( no   | kein  ) = -1
p( did not |hat kein)=-0.5
p( has no  |hat kein ) = -1.5
p( goal| Tor) = -2
p( goals|Tor ) = -3
p( shot  |geschossen ) = -1
p( scored|geschossen ) = -2
p( shoot |geschossen ) = -1
p(scored .|Tor geschossen .)=-10
p(.|.) = -0.5
```

	Peter	*has*	*not-a*	*goal*	*scored*	*.*	
-1	0	1	2	3	4	5	6
	Peter	**hat**	**kein**	**Tor**	**geschossen**	**.**	

Language Model

ARPA format

```
-0.5  .                -3
-4    goal             -0.5
-2    not              -1
-1    Peter
-6    scored           -3
-5    shoot            -0.1
-0.5  . </s>
-2    <s> Peter        -1
-3    <s> has          -0.5
-1    <s> did          -0.7
-0.5  did not          -0.5
-1.5  has no           -2
-2    shoot goals      -1
-1    <s> Peter did
-1    not shoot goals
-0.5  Peter did not
```



Decoding Example

Translation Model

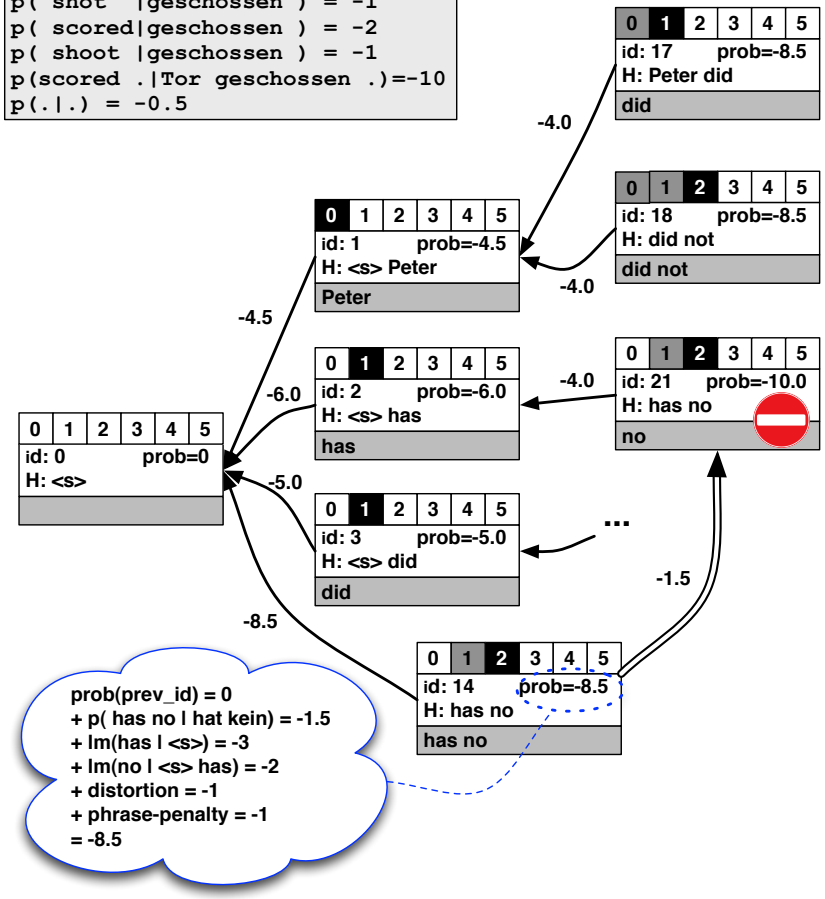
```
p( Peter| Peter ) = -1.5
p( has  | hat   ) = -1
p( did  | hat   ) = -2
p( no   | kein  ) = -1
p( did not |hat kein)=-0.5
p( has no  |hat kein ) = -1.5
p( goal| Tor) = -2
p( goals|Tor ) = -3
p( shot  |geschossen ) = -1
p( scored|geschossen ) = -2
p( shoot |geschossen ) = -1
p(scored .|Tor geschossen .)=-10
p(.|.) = -0.5
```

	Peter	*has*	*not-a*	*goal*	*scored*	*.*	
-1	0	1	2	3	4	5	6
	Peter	**hat**	**kein**	**Tor**	**geschossen**	**.**	

Language Model

ARPA format

```
-0.5  .                -3
-4    goal             -0.5
-2    not              -1
-1    Peter
-6    scored           -3
-5    shoot            -0.1
-0.5  . </s>
-2    <s> Peter        -1
-3    <s> has          -0.5
-1    <s> did          -0.7
-0.5  did not          -0.5
-1.5  has no           -2
-2    shoot goals      -1
-1    <s> Peter did
-1    not shoot goals
-0.5  Peter did not
```



Decoding Example

Translation Model

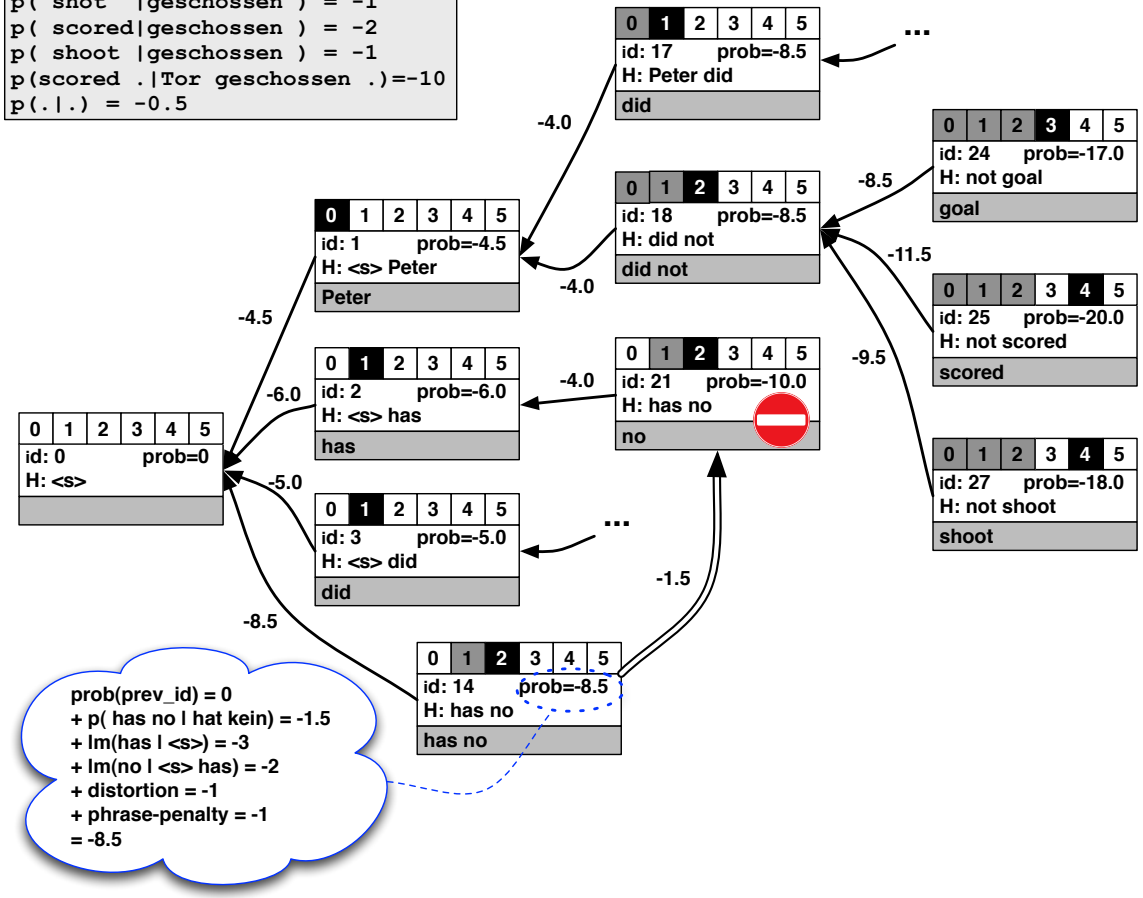
```
p( Peter| Peter ) = -1.5
p( has  | hat   ) = -1
p( did  | hat   ) = -2
p( no   | kein  ) = -1
p( did not |hat kein)=-0.5
p( has no  |hat kein ) = -1.5
p( goal| Tor) = -2
p( goals|Tor ) = -3
p( shot  |geschossen ) = -1
p( scored|geschossen ) = -2
p( shoot |geschossen ) = -1
p(scored .|Tor geschossen .)=-10
p(.|.) = -0.5
```

	Peter	*has*	*not-a*	*goal*	*scored*	*.*	
-1	0	1	2	3	4	5	6
	Peter	**hat**	**kein**	**Tor**	**geschossen**	**.**	

Language Model

ARPA format

```
-0.5  .                -3
-4    goal             -0.5
-2    not              -1
-1    Peter
-6    scored           -3
-5    shoot            -0.1
-0.5  . </s>
-2    <s> Peter        -1
-3    <s> has          -0.5
-1    <s> did          -0.7
-0.5  did not          -0.5
-1.5  has no           -2
-2    shoot goals      -1
-1    <s> Peter did
-1    not shoot goals
-0.5  Peter did not
```

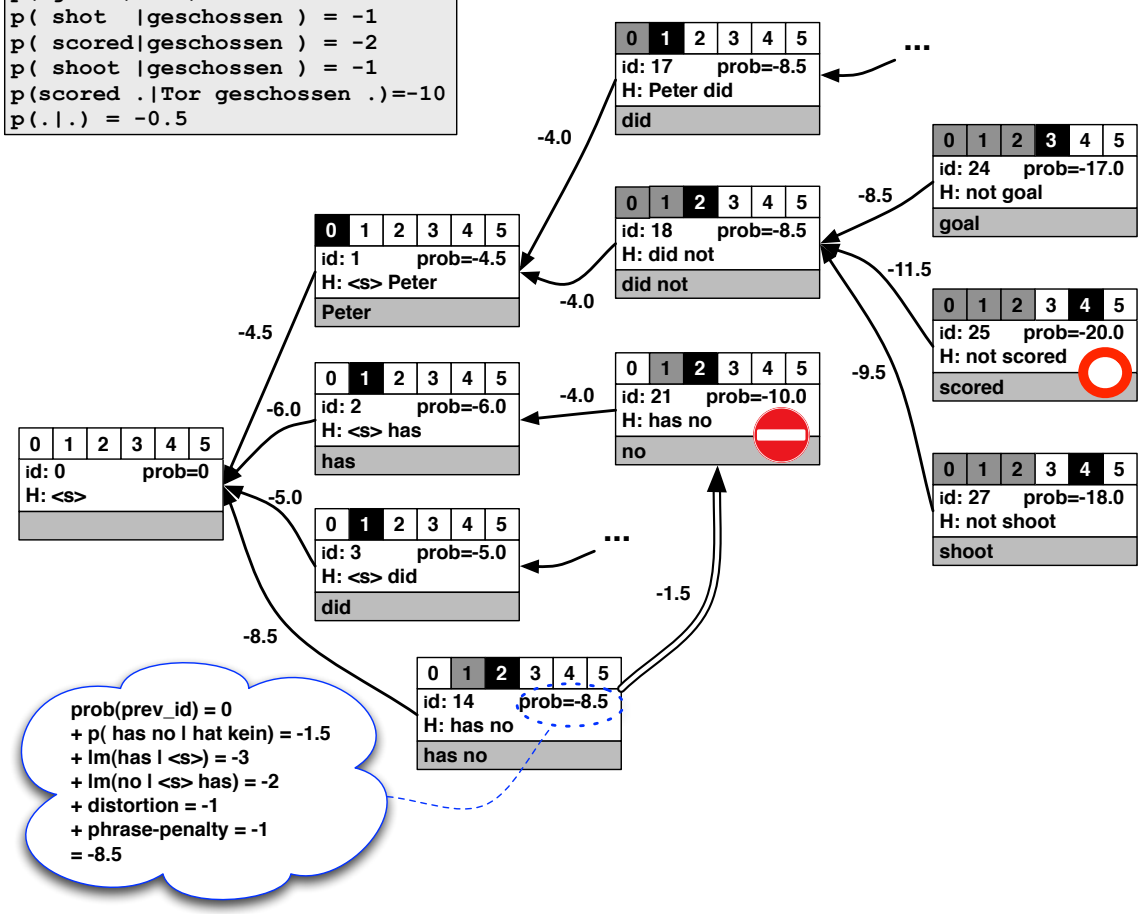


Decoding Example

Translation Model

```
p( Peter| Peter ) = -1.5
p( has  | hat   ) = -1
p( did  | hat   ) = -2
p( no   | kein  ) = -1
p( did not |hat kein)=-0.5
p( has no  |hat kein ) = -1.5
p( goal| Tor) = -2
p( goals|Tor ) = -3
p( shot  |geschossen ) = -1
p( scored|geschossen ) = -2
p( shoot |geschossen ) = -1
p(scored .|Tor geschossen .)=-10
p(.|.) = -0.5
```

	Peter	*has*	*not-a*	*goal*	*scored*	*.*	
-1	0	1	2	3	4	5	6
	Peter	**hat**	**kein**	**Tor**	**geschossen**	**.**	



Language Model

ARPA format

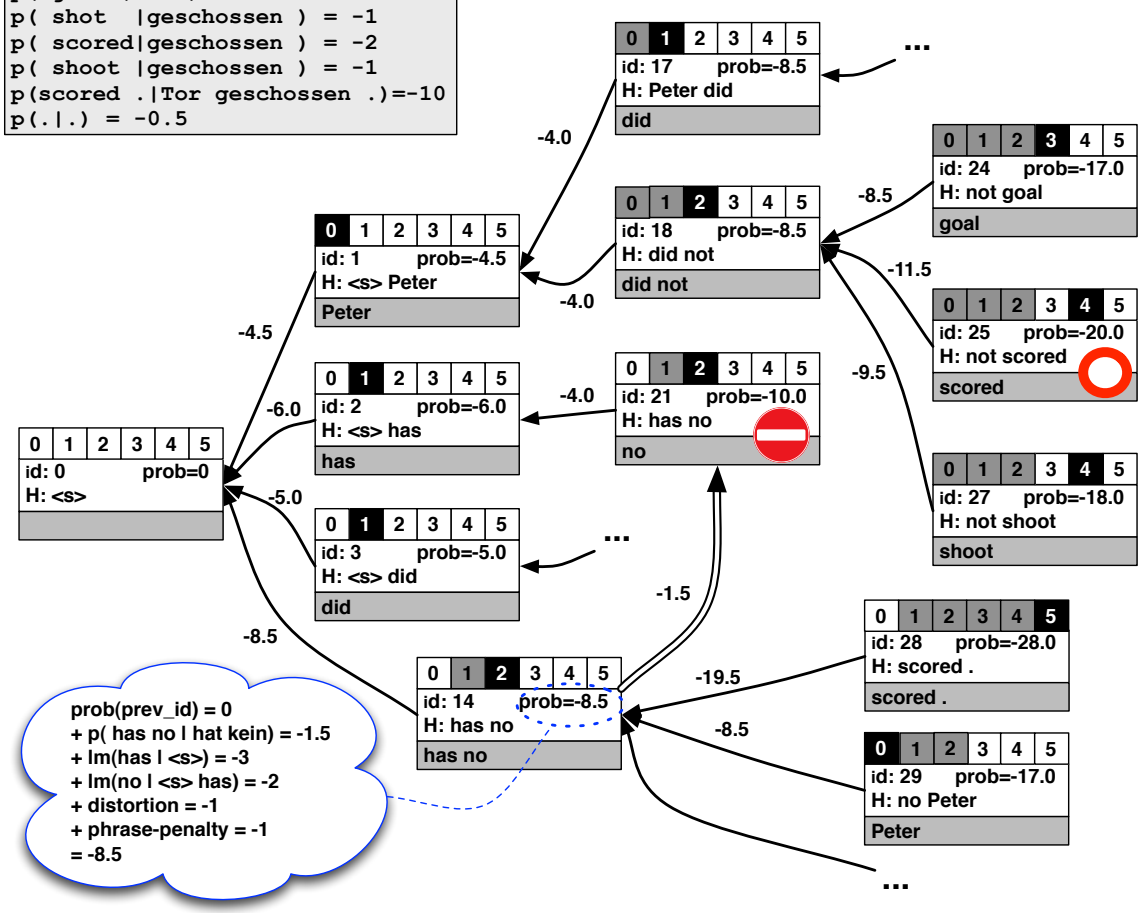
```
-0.5  .                -3
-4    goal             -0.5
-2    not              -1
-1    Peter
-6    scored           -3
-5    shoot            -0.1
-0.5  . </s>
-2    <s> Peter        -1
-3    <s> has          -0.5
-1    <s> did          -0.7
-0.5  did not          -0.5
-1.5  has no           -2
-2    shoot goals      -1
-1    <s> Peter did
-1    not shoot goals
-0.5  Peter did not
```

Decoding Example

Translation Model

```
p( Peter| Peter ) = -1.5
p( has  | hat   ) = -1
p( did  | hat   ) = -2
p( no   | kein  ) = -1
p( did not |hat kein)=-0.5
p( has no  |hat kein ) = -1.5
p( goal| Tor) = -2
p( goals|Tor ) = -3
p( shot  |geschossen ) = -1
p( scored|geschossen ) = -2
p( shoot |geschossen ) = -1
p(scored .|Tor geschossen .)=-10
p(.|.) = -0.5
```

	Peter	*has*	*not-a*	*goal*	*scored*	*.*	
-1	0	1	2	3	4	5	6
	Peter	**hat**	**kein**	**Tor**	**geschossen**	**.**	



Language Model

ARPA format

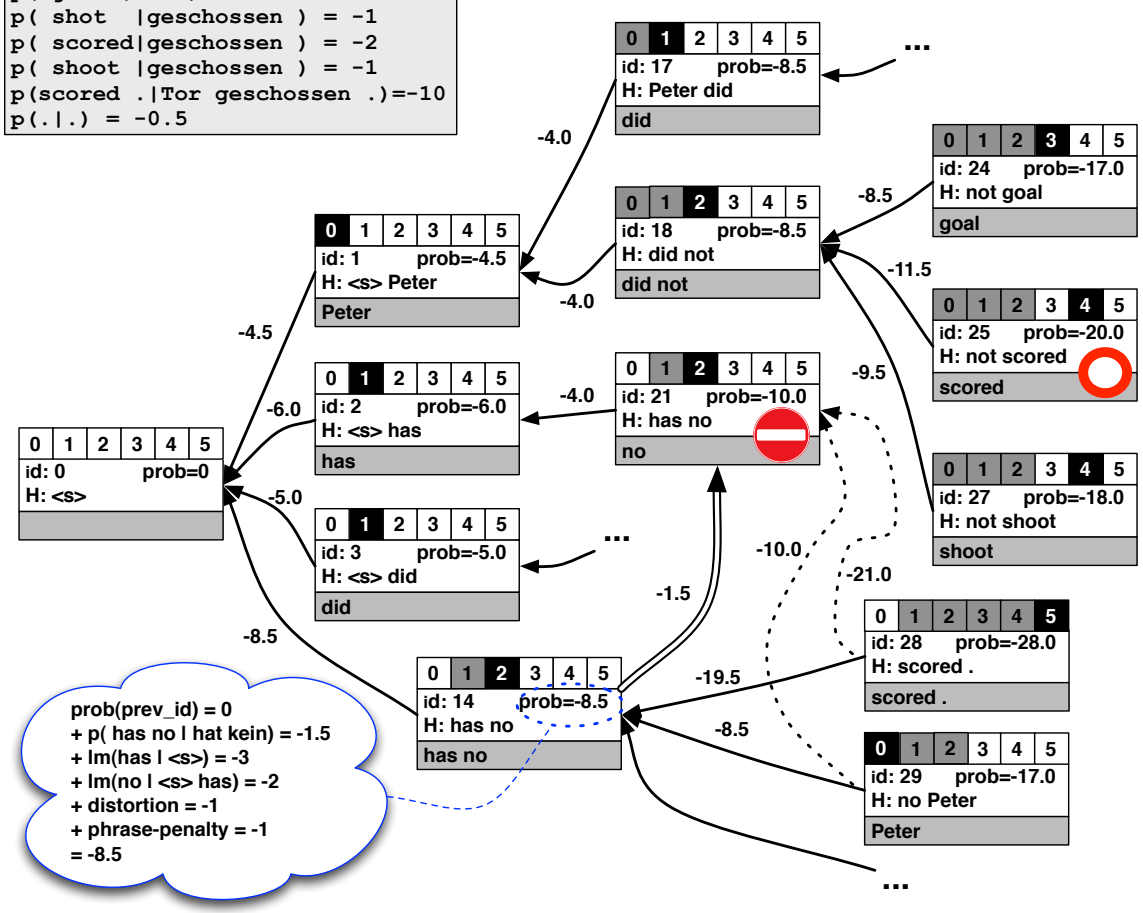
```
-0.5   .                -3
-4     goal             -0.5
-2     not              -1
-1     Peter
-6     scored           -3
-5     shoot            -0.1
-0.5   . </s>
-2     <s> Peter        -1
-3     <s> has          -0.5
-1     <s> did          -0.7
-0.5   did not          -0.5
-1.5   has no           -2
-2     shoot goals      -1
-1     <s> Peter did
-1     not shoot goals
-0.5   Peter did not
```

Decoding Example

Translation Model

```
p( Peter| Peter ) = -1.5
p( has  | hat   ) = -1
p( did  | hat   ) = -2
p( no   | kein  ) = -1
p( did not |hat kein)=-0.5
p( has no  |hat kein ) = -1.5
p( goal| Tor) = -2
p( goals|Tor ) = -3
p( shot  |geschossen ) = -1
p( scored|geschossen ) = -2
p( shoot |geschossen ) = -1
p(scored .|Tor geschossen .)=-10
p(.|.) = -0.5
```

	Peter	*has*	*not-a*	*goal*	*scored*	*.*	
-1	0	1	2	3	4	5	6
	Peter	**hat**	**kein**	**Tor**	**geschossen**	**.**	



Language Model

ARPA format

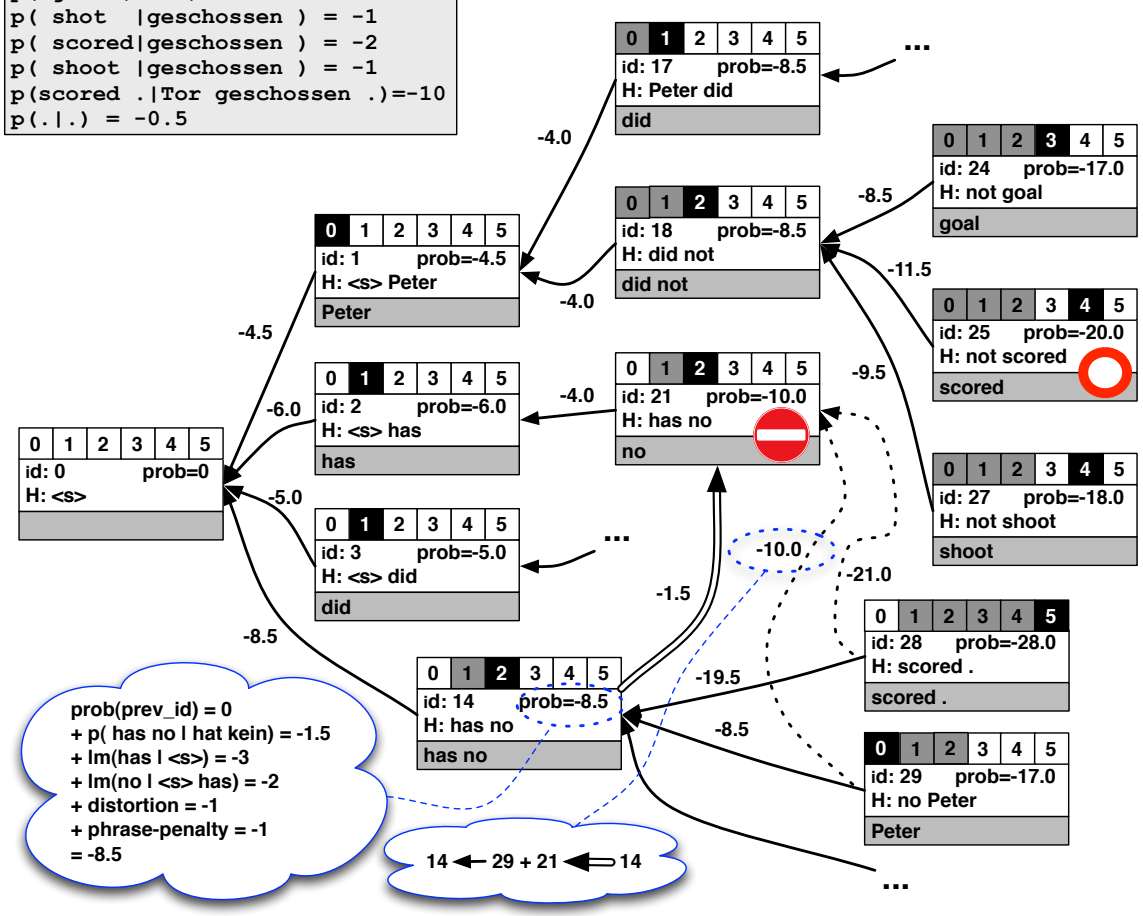
```
-0.5  .              -3
-4    goal           -0.5
-2    not            -1
-1    Peter
-6    scored         -3
-5    shoot          -0.1
-0.5  . </s>
-2    <s> Peter      -1
-3    <s> has        -0.5
-1    <s> did        -0.7
-0.5  did not        -0.5
-1.5  has no         -2
-2    shoot goals    -1
-1    <s> Peter did
-1    not shoot goals
-0.5  Peter did not
```

Decoding Example

Translation Model

```
p( Peter| Peter ) = -1.5
p( has  | hat  ) = -1
p( did  | hat  ) = -2
p( no   | kein ) = -1
p( did not |hat kein)=-0.5
p( has no  |hat kein ) = -1.5
p( goal| Tor) = -2
p( goals|Tor ) = -3
p( shot  |geschossen ) = -1
p( scored|geschossen ) = -2
p( shoot |geschossen ) = -1
p(scored .|Tor geschossen .)=-10
p(.|.) = -0.5
```

	Peter	*has*	*not-a*	*goal*	*scored*	*.*	
-1	0	1	2	3	4	5	6
	Peter	**hat**	**kein**	**Tor**	**geschossen**	**.**	



Language Model

ARPA format

```
-0.5   .                -3
-4     goal             -0.5
-2     not              -1
-1     Peter
-6     scored           -3
-5     shoot            -0.1
-0.5   . </s>
-2     <s> Peter        -1
-3     <s> has          -0.5
-1     <s> did          -0.7
-0.5   did not          -0.5
-1.5   has no           -2
-2     shoot goals      -1
-1     <s> Peter did
-1     not shoot goals
-0.5   Peter did not
```

Decoding Example

Translation Model

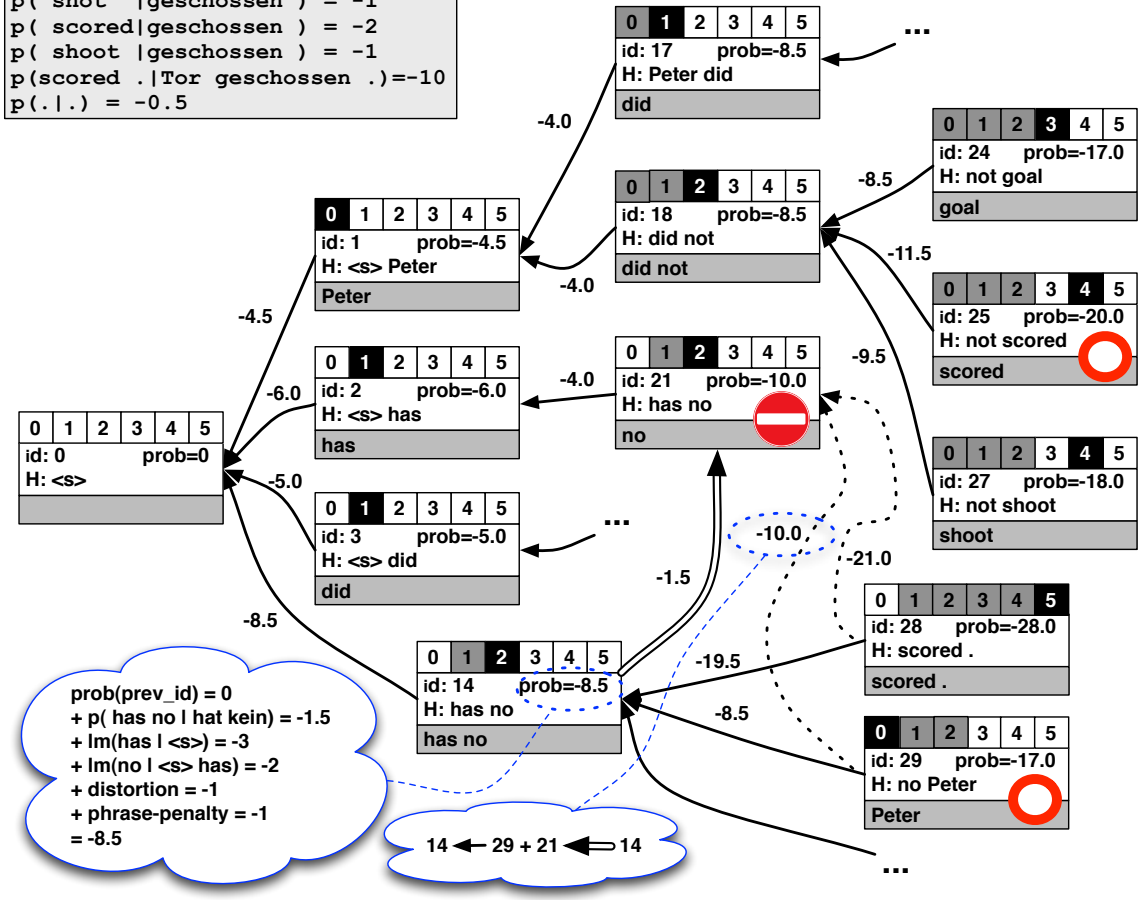
```
p( Peter| Peter ) = -1.5
p( has  | hat   ) = -1
p( did  | hat   ) = -2
p( no   | kein  ) = -1
p( did not |hat kein)=-0.5
p( has no  |hat kein ) = -1.5
p( goal| Tor) = -2
p( goals|Tor ) = -3
p( shot  |geschossen ) = -1
p( scored|geschossen ) = -2
p( shoot |geschossen ) = -1
p(scored .|Tor geschossen .)=-10
p(.|.) = -0.5
```

	Peter	*has*	*not-a*	*goal*	*scored*	*.*	
-1	0	1	2	3	4	5	6
	Peter	**hat**	**kein**	**Tor**	**geschossen**	**.**	

Language Model

ARPA format

```
-0.5  .                -3
-4    goal             -0.5
-2    not              -1
-1    Peter
-6    scored           -3
-5    shoot            -0.1
-0.5  . </s>
-2    <s> Peter        -1
-3    <s> has          -0.5
-1    <s> did          -0.7
-0.5  did not          -0.5
-1.5  has no           -2
-2    shoot goals      -1
-1    <s> Peter did
-1    not shoot goals
-0.5  Peter did not
```



Decoding Example

Translation Model

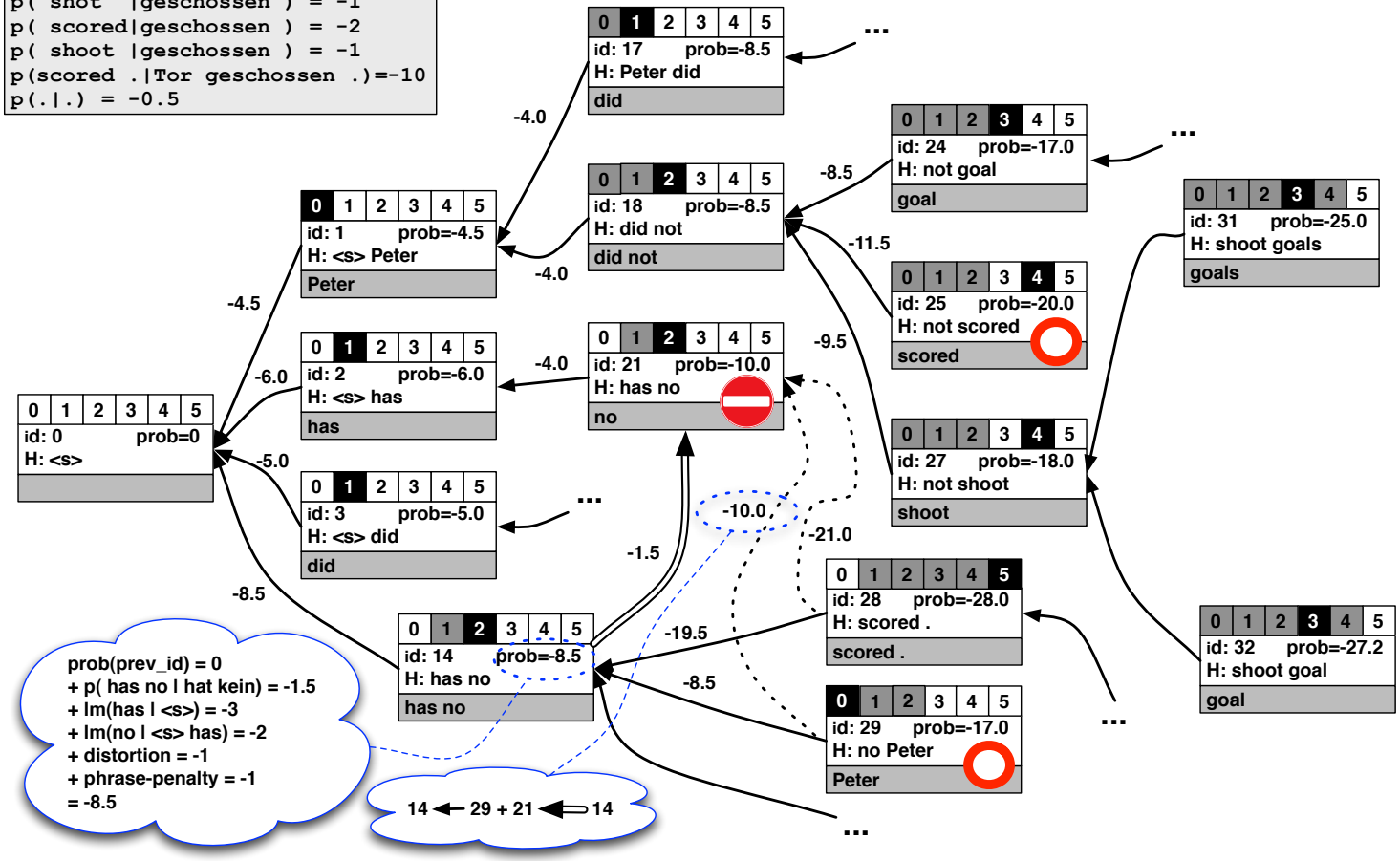
```
p( Peter| Peter ) = -1.5
p( has  | hat  ) = -1
p( did  | hat  ) = -2
p( no   | kein ) = -1
p( did not |hat kein)=-0.5
p( has no  |hat kein ) = -1.5
p( goal| Tor) = -2
p( goals|Tor ) = -3
p( shot  |geschossen ) = -1
p( scored|geschossen ) = -2
p( shoot |geschossen ) = -1
p(scored .|Tor geschossen .)=-10
p(.|.) = -0.5
```

	Peter	*has*	*not-a*	*goal*	*scored*	*.*	
-1	0	1	2	3	4	5	6
	Peter	**hat**	**kein**	**Tor**	**geschossen**	**.**	

Language Model

ARPA format

```
-0.5  .                -3
-4    goal             -0.5
-2    not              -1
-1    Peter
-6    scored           -3
-5    shoot            -0.1
-0.5  . </s>
-2    <s> Peter        -1
-3    <s> has          -0.5
-1    <s> did          -0.7
-0.5  did not          -0.5
-1.5  has no           -2
-2    shoot goals      -1
-1    <s> Peter did
-1    not shoot goals
-0.5  Peter did not
```



Decoding Example

Translation Model

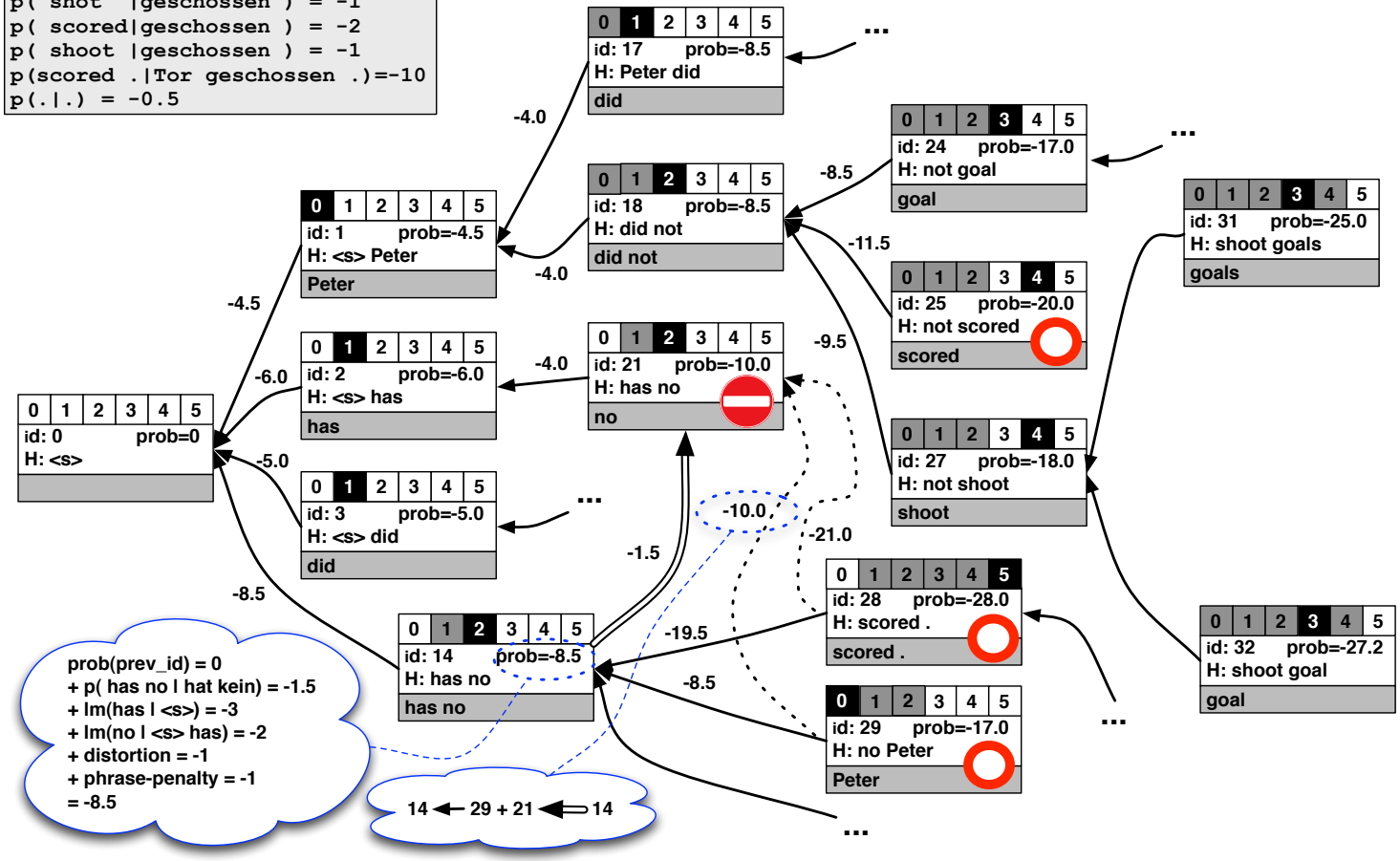
```
p( Peter| Peter ) = -1.5
p( has  | hat   ) = -1
p( did  | hat   ) = -2
p( no   | kein  ) = -1
p( did not |hat kein)=-0.5
p( has no  |hat kein ) = -1.5
p( goal| Tor) = -2
p( goals|Tor ) = -3
p( shot  |geschossen ) = -1
p( scored|geschossen ) = -2
p( shoot |geschossen ) = -1
p(scored .|Tor geschossen .)=-10
p(.|.) = -0.5
```

	Peter	*has*	*not-a*	*goal*	*scored*	*.*	
-1	0	1	2	3	4	5	6
	Peter	**hat**	**kein**	**Tor**	**geschossen**	**.**	

Language Model

ARPA format

```
-0.5   .                -3
-4     goal             -0.5
-2     not              -1
-1     Peter
-6     scored           -3
-5     shoot            -0.1
-0.5   . </s>
-2     <s> Peter        -1
-3     <s> has          -0.5
-1     <s> did          -0.7
-0.5   did not          -0.5
-1.5   has no           -2
-2     shoot goals      -1
-1     <s> Peter did
-1     not shoot goals
-0.5   Peter did not
```



Decoding Example

Translation Model

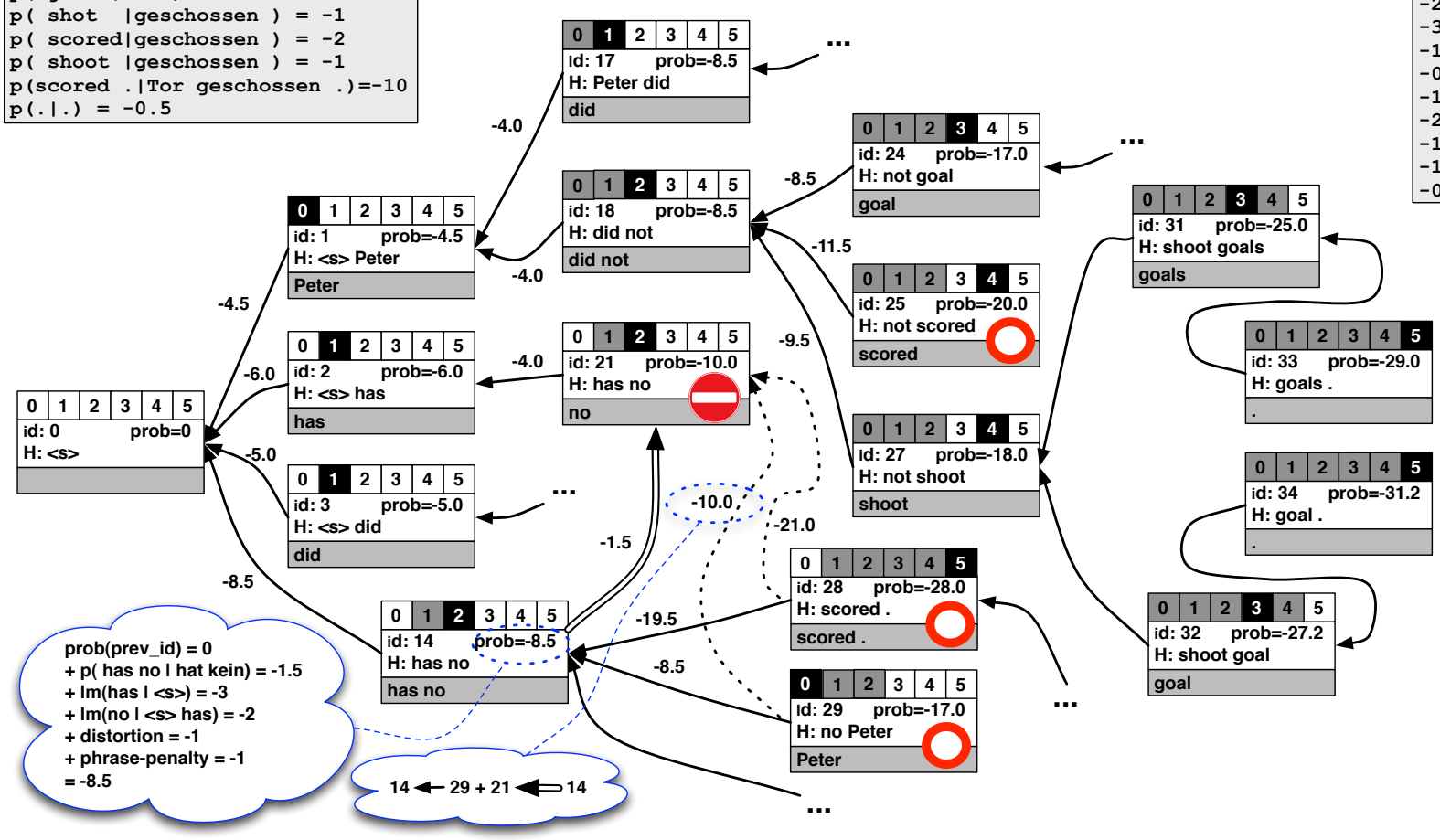
```
p( Peter| Peter ) = -1.5
p( has  | hat   ) = -1
p( did  | hat   ) = -2
p( no   | kein  ) = -1
p( did not |hat kein)=-0.5
p( has no  |hat kein ) = -1.5
p( goal| Tor) = -2
p( goals|Tor ) = -3
p( shot  |geschossen ) = -1
p( scored|geschossen ) = -2
p( shoot |geschossen ) = -1
p(scored .|Tor geschossen .)=-10
p(.|.) = -0.5
```

	Peter	*has*	*not-a*	*goal*	*scored*	*.*	
-1	0	1	2	3	4	5	6
	Peter	**hat**	**kein**	**Tor**	**geschossen**	**.**	

Language Model

ARPA format

```
-0.5  .                -3
-4    goal             -0.5
-2    not              -1
-1    Peter
-6    scored           -3
-5    shoot            -0.1
-0.5  . </s>
-2    <s> Peter        -1
-3    <s> has          -0.5
-1    <s> did          -0.7
-0.5  did not          -0.5
-1.5  has no           -2
-2    shoot goals      -1
-1    <s> Peter did
-1    not shoot goals
-0.5  Peter did not
```



Decoding Example

Translation Model

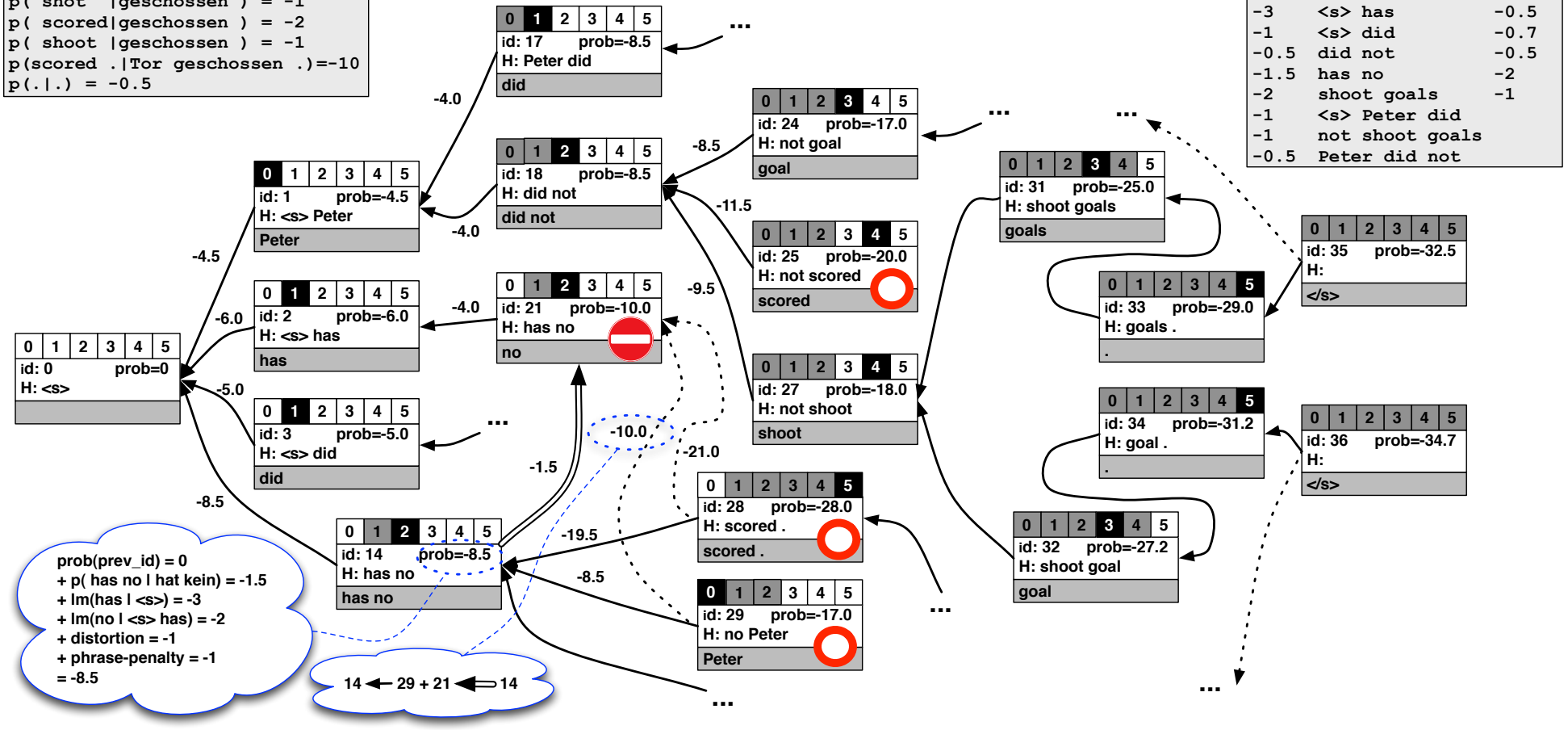
```
p( Peter| Peter ) = -1.5
p( has  | hat   ) = -1
p( did  | hat   ) = -2
p( no   | kein  ) = -1
p( did not |hat kein)=-0.5
p( has no  |hat kein ) = -1.5
p( goal| Tor) = -2
p( goals|Tor ) = -3
p( shot  |geschossen ) = -1
p( scored|geschossen ) = -2
p( shoot |geschossen ) = -1
p(scored .|Tor geschossen .)=-10
p(.|.) = -0.5
```

	Peter	*has*	*not-a*	*goal*	*scored*	*.*	
-1	0	1	2	3	4	5	6
	Peter	**hat**	**kein**	**Tor**	**geschossen**	**.**	

Language Model

ARPA format

```
-0.5  .                -3
-4    goal             -0.5
-2    not              -1
-1    Peter
-6    scored           -3
-5    shoot            -0.1
-0.5  . </s>
-2    <s> Peter        -1
-3    <s> has          -0.5
-1    <s> did          -0.7
-0.5  did not          -0.5
-1.5  has no           -2
-2    shoot goals      -1
-1    <s> Peter did
-1    not shoot goals
-0.5  Peter did not
```



Decoding Example

Translation Model

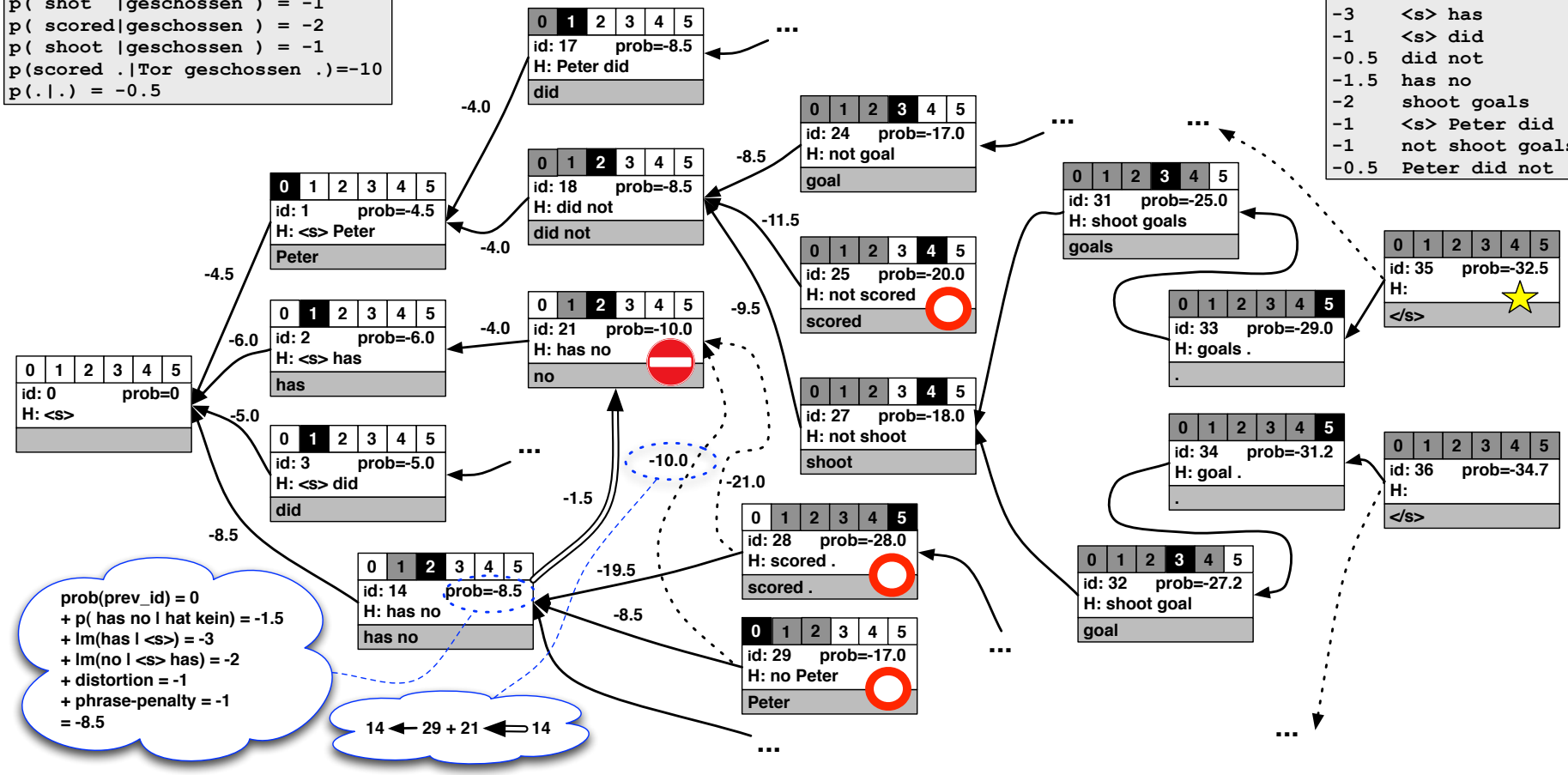
```
p( Peter| Peter ) = -1.5
p( has  | hat   ) = -1
p( did  | hat   ) = -2
p( no   | kein  ) = -1
p( did not |hat kein)=-0.5
p( has no  |hat kein ) = -1.5
p( goal| Tor) = -2
p( goals|Tor ) = -3
p( shot  |geschossen ) = -1
p( scored|geschossen ) = -2
p( shoot |geschossen ) = -1
p(scored .|Tor geschossen .)=-10
p(.|.) = -0.5
```

	Peter	*has*	*not-a*	*goal*	*scored*	*.*	
-1	0	1	2	3	4	5	6
	Peter	**hat**	**kein**	**Tor**	**geschossen**	**.**	

Language Model

ARPA format

```
-0.5  .                -3
-4    goal             -0.5
-2    not              -1
-1    Peter
-6    scored           -3
-5    shoot            -0.1
-0.5  . </s>
-2    <s> Peter        -1
-3    <s> has          -0.5
-1    <s> did          -0.7
-0.5  did not          -0.5
-1.5  has no           -2
-2    shoot goals      -1
-1    <s> Peter did
-1    not shoot goals
-0.5  Peter did not
```



Decoding Example

Translation Model

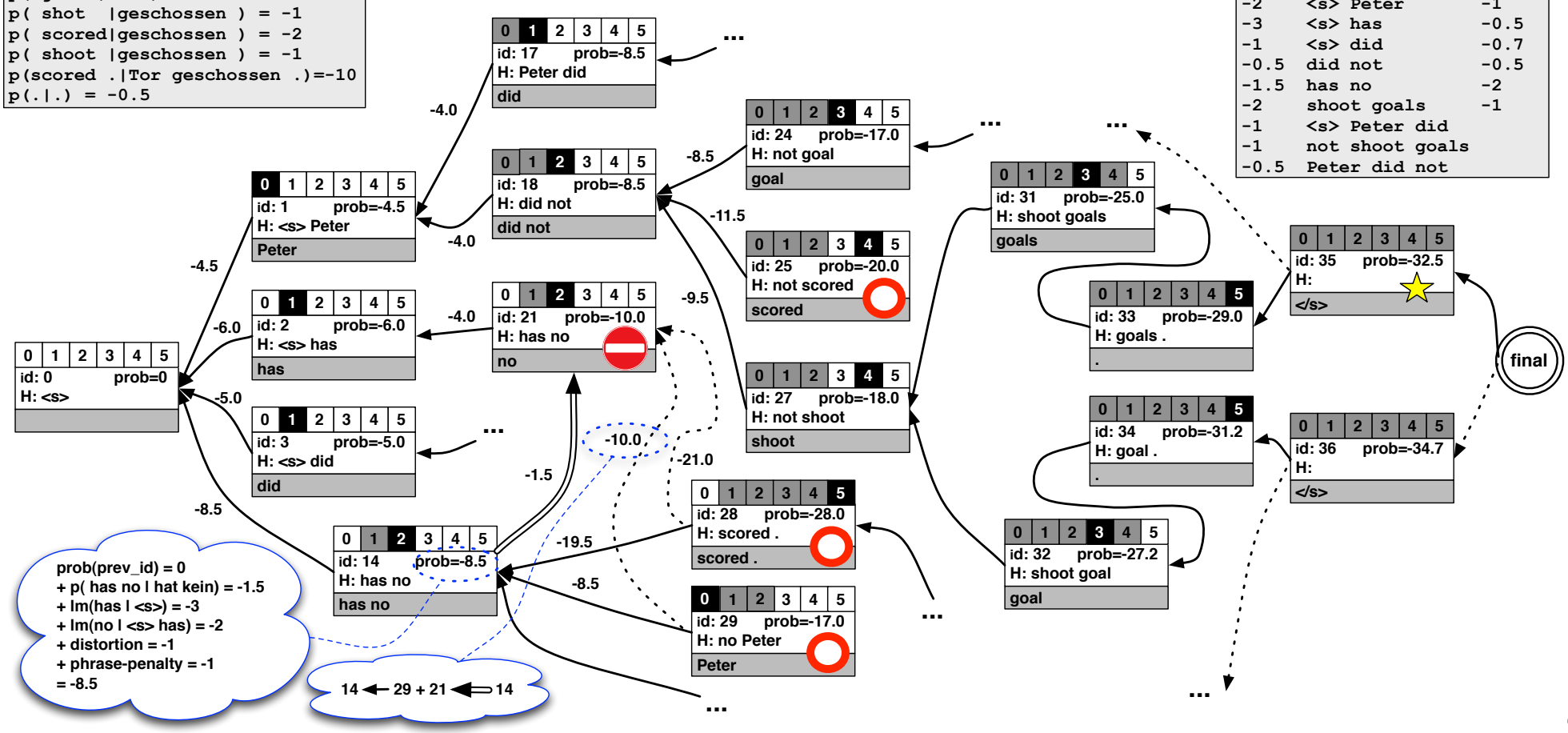
```
p( Peter| Peter ) = -1.5
p( has  | hat   ) = -1
p( did  | hat   ) = -2
p( no   | kein  ) = -1
p( did not |hat kein)=-0.5
p( has no  |hat kein ) = -1.5
p( goal| Tor) = -2
p( goals|Tor ) = -3
p( shot  |geschossen ) = -1
p( scored|geschossen ) = -2
p( shoot |geschossen ) = -1
p(scored .|Tor geschossen .)=-10
p(.|.) = -0.5
```

	Peter	*has*	*not-a*	*goal*	*scored*	*.*	
-1	0	1	2	3	4	5	6
	Peter	**hat**	**kein**	**Tor**	**geschossen**	**.**	

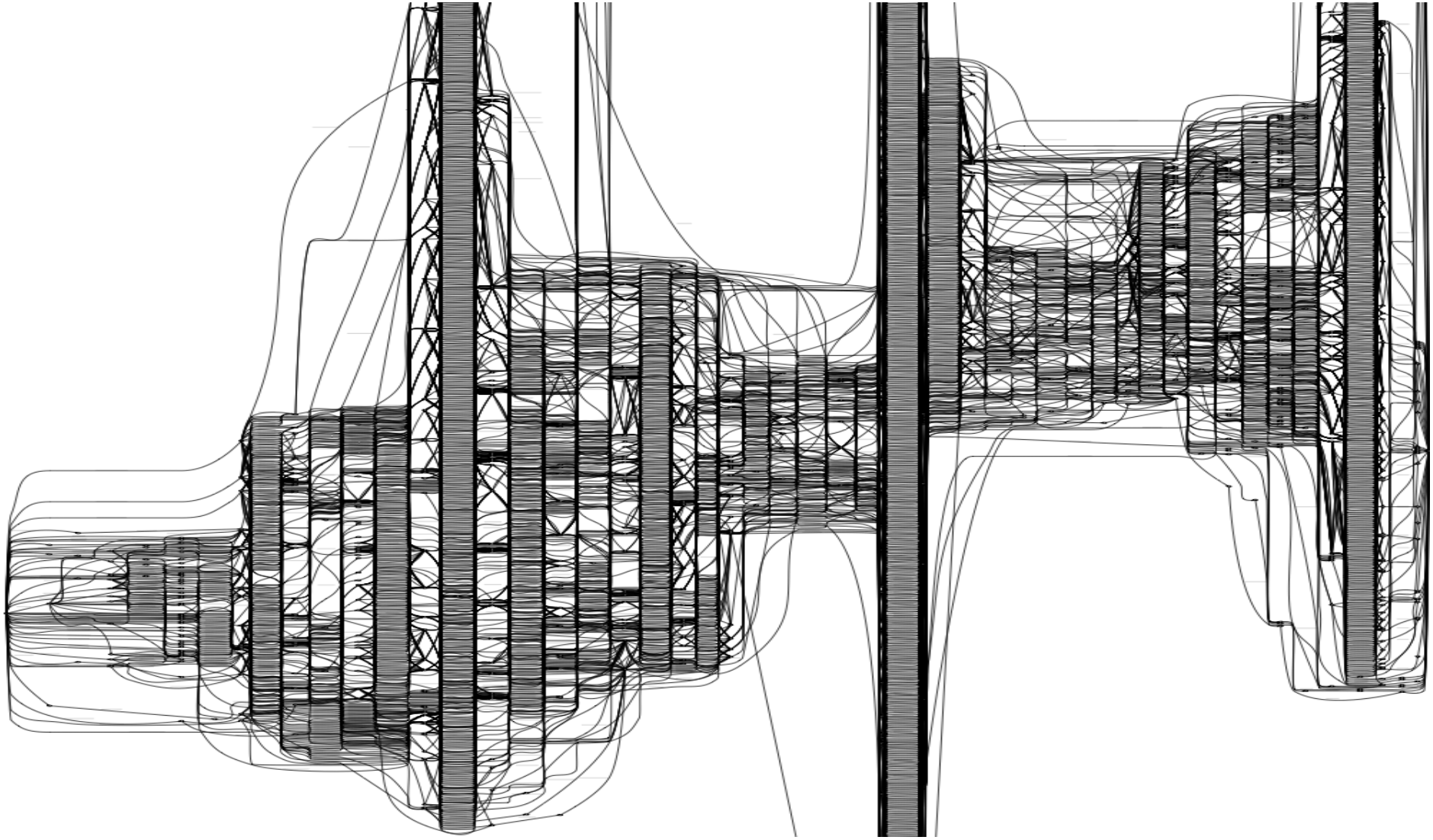
Language Model

ARPA format

```
-0.5  .              -3
-4    goal           -0.5
-2    not            -1
-1    Peter
-6    scored         -3
-5    shoot          -0.1
-0.5  . </s>
-2    <s> Peter      -1
-3    <s> has        -0.5
-1    <s> did        -0.7
-0.5  did not        -0.5
-1.5  has no         -2
-2    shoot goals    -1
-1    <s> Peter did
-1    not shoot goals
-0.5  Peter did not
```



. . . and Here's the Real Thing



Viterbi Translation

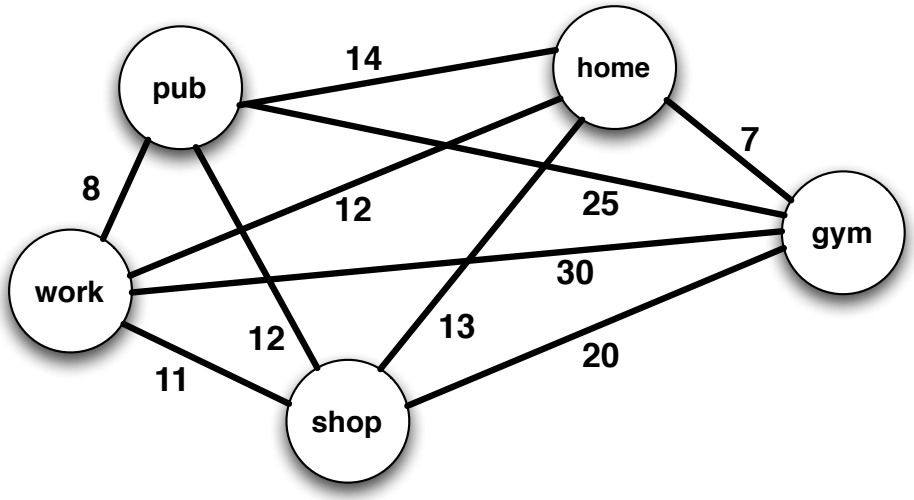
- Given the lattice L that we have just generated,
  - $\tilde{E}$ is the set of all paths in L
- The Viterbi path e_{vit} of L is the path with the highest probability,
  - I.e., the path with the highest sum of log-prob transition probabilities
  - This can be easily extracted from the lattice by following the back-pointers from the final state until we reach the initial state
    - Note, this generates the translation in right-to-left order and needs to be reversed
- As all transition probabilities are based on the model probabilities, this is equal to

$$e_{vit} = \arg\max_{e \in \tilde{E}} \exp\left(\sum_{m=1}^{M} \lambda_m h_m(e, f)\right)$$

Decoding Complexity

- Some complexities:
  - Average number of phrase translation per source phrase: $\frac{1}{\sum_{\bar{e}} n_{1+}(\bar{e}|\bullet)} \sum_{\bar{f}} n_{1+}(\bullet|\bar{f})$
  - Number of phrase segmentations for a sentence of length $n = 2^{n-1}$
  - Number of reorderings for a sentence of length $n = n!$
- (Very) naïve and unrestricted decoding complexity of $O(\frac{1}{\sum_{\bar{e}} n_{1+}(\bar{e}|\bullet)} \sum_{\bar{f}} n_{1+}(\bullet|\bar{f}) \cdot 2^{n-1} \cdot n!)$
- Of course, many of the decoding paths share sub-paths, which should reduce complexity
- Let's focus on a very simplified version of SMT with deterministic word translation and word reordering only

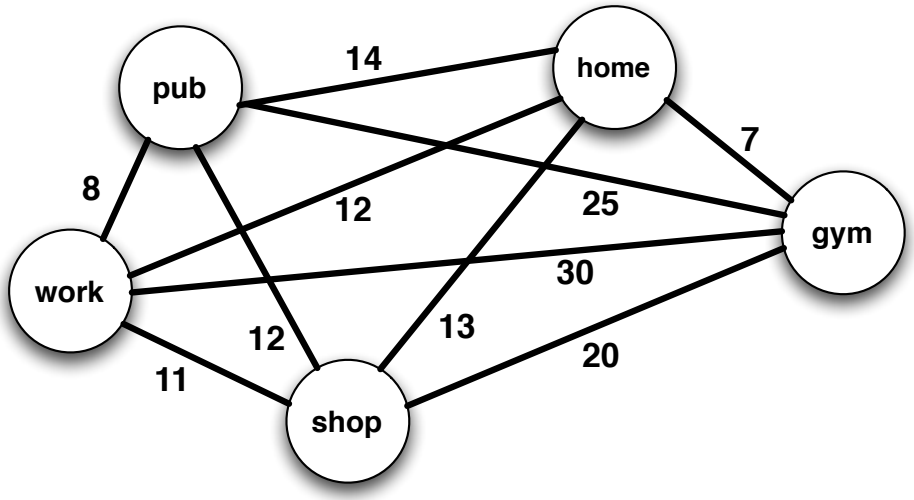
Decoding Complexity

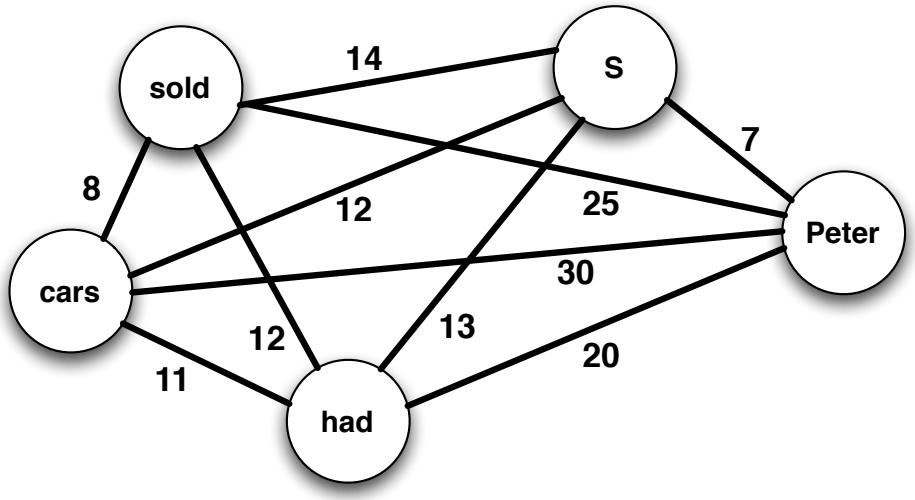


$d(\text{loc}_1, \text{loc}_2) = \text{walking distance}$

- Traveling Salesman Problem: One of the hardest problems in combinatorial optimization (NP complete)
  - Brute-force: $O(n!)$ where $n =$ number of states (modulo start/end state)
  - Held & Karp (1962): Dynamic programming solution: $O(n^2 2^n)$

Decoding Complexity





$d(\text{loc}_1, \text{loc}_2) = \text{walking distance}$

$d(w_1, w_2) = -\log(p(w_1|w_2) + p(w_2|w_1))$

- Traveling Salesman Problem: One of the hardest problems in combinatorial optimization (NP complete)
  - Brute-force: $O(n!)$ where $n =$ number of states (modulo start/end state)
  - Held & Karp (1962): Dynamic programming solution: $O(n^2 2^n)$
- TSP can be reduced to SMT problem
  - Here: Each node is a foreign word with one translation and a symmetric (for illustration purposes) language model

Decoding Complexity

- The brute force SMT decoding complexity is $O(n!)$, where n is the number of tokens in a foreign sentence
- In order to reduce this we consider a number of strategies:

  - **Dynamic programming methods:** reducing it to $O(n^2 2^n)$, which is considerably less complex than $O(n!)$ for larger values of n
  - **Heuristic Search Space Constraints:** hard-coded restrictions of certain hypothesis expansions
  - **Beam-Search Pruning:** eliminating states that are unlikely to achieve higher probabilities than the best comparable state

Dynamic Programming

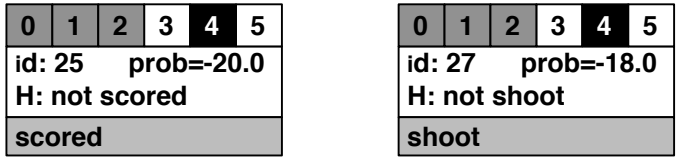
- A complex computational problem can be solved with dynamic programming (DP) if the solution to the general problem can be based on the solutions to it subproblems s.t.
  - There is an ordering on the subproblems, and a relation that shows how to solve a subproblem given the answers to 'smaller' subproblems, that is, subproblems that appear earlier in the ordering
- DP is commonly used in Computer Science, examples include
  - String edit distance (Levenshtein distance)
  - Longest increasing subsequence problems
  - Balanced partitioning

Dynamic Programming

- Let's say a problem P covers items $1 \ldots n$: $P[1,n]$
  - If the nature problem is consistent with the DP conditions
  - and we memorized the solutions for the two subproblems $P[2,n]$ and $P[1,n-1]$
  - then we also know that the solution for $P[1,n]$ is $\arg\max\{P[1,n-1] \otimes P[n,n], P[1,1] \otimes P[2,n]\}$
- How do we obtain the solution for the subproblems, e.g., $P[1,n-1]$?
  - well, we memorized the solutions for the two subproblems $P[2,n-1]$ and $P[1,n-2]$, then ...
- Initialize minimal subproblems that can not be broken down any further, such as $P[i,i]$ for $1 \leq i \leq n$
- The complexity of a DP solution depends on the number of subproblems that need to be computed
  - Here: $\forall_{1 \leq i \leq j \leq n} : P[i,j]$, i.e., $\frac{n^2+n}{2}$ subproblems, so $O(n^2)$

DP Decoding

- How do we define a sub-problem in SMT decoding?
- First attempt: Two states solve the same subproblem if they cover the same foreign positions, i.e., if their coverage vectors are identical
- In unrestricted form: $O(2^k)$, where k is the number of source words
- Problem:



  - State `id:27` would be better solution for this subproblem
  - Both states have different $n-1$-gram histories
    - Combining it with a solution for position 3 or 5 might prefer `id:25` due to language model scores

DP Decoding

- Second attempt: Two states solve the same subproblem if
  - *(a)* they cover the same foreign positions, i.e., if their coverage vectors are identical
  - *(b)* the last translated foreign position values are identical
  - *(c)* their $n-1$-gram language model histories are identical
- Condition
  - *(a)* ensures that both states have solve translation subproblems of similar difficulty
  - *(b)* ensures that any reordering combined with subproblem incurs the same distortion cost
  - *(c)* ensures that any translation continuation combined with subproblem results in the same language model probability
- Additional features (e.g., lexicalized reordering) can introduce further constraints

Recombination

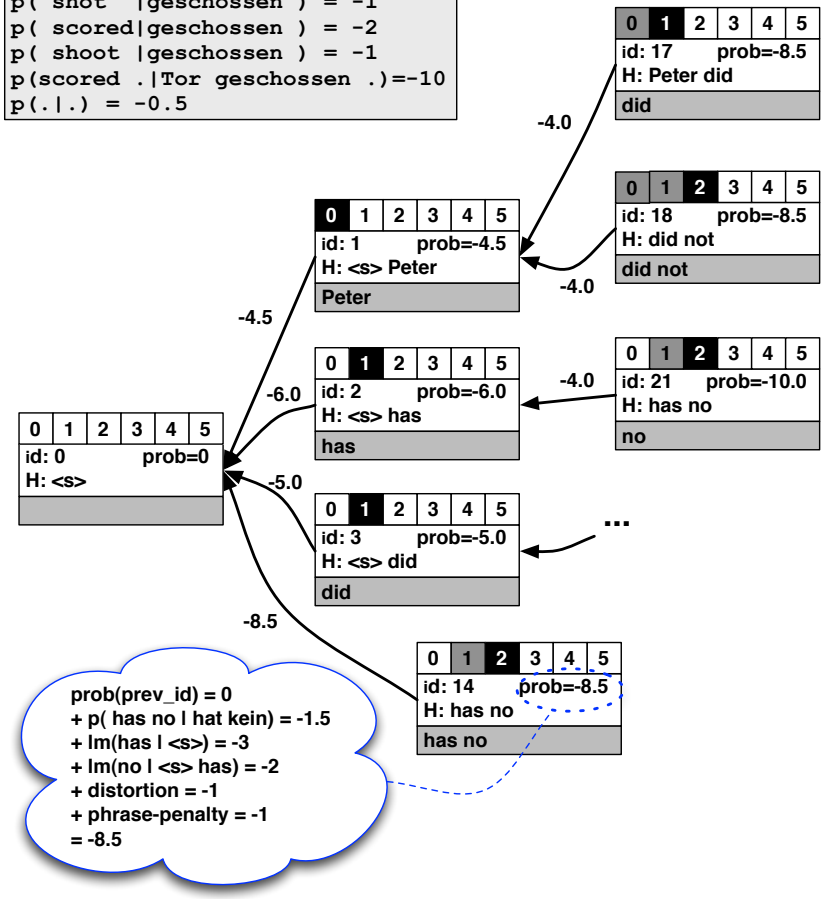
- If we are just interested in finding the 1-best, Viterbi translation of a foreign sentence, DP decoding is sufficient
  - If two states solve the same subproblem (as defined in conditions a–c), we can safely remove the lower-scoring state from our search space
- If we are interested in finding the n-best translations we
  - exclude the lower scoring state from any further expansions
  - link states that solve the same subproblem to the optimal solution state for this subproblem (recombination)
- State transitions from the optimal state link back to the recombined solutions (with adjusted transition costs)
  - Add the cost difference between optimal and recombined solution
- Linking allows us to follow the best sub-optimal path during back-tracing

Recombination Example

Translation Model

```
p( Peter| Peter ) = -1.5
p( has  | hat   ) = -1
p( did  | hat   ) = -2
p( no   | kein  ) = -1
p( did not |hat kein)=-0.5
p( has no  |hat kein ) = -1.5
p( goal| Tor) = -2
p( goals|Tor ) = -3
p( shot  |geschossen ) = -1
p( scored|geschossen ) = -2
p( shoot |geschossen ) = -1
p(scored .|Tor geschossen .)=-10
p(.|.) = -0.5
```

	Peter	*has*	*not-a*	*goal*	*scored*	*.*	
-1	0	1	2	3	4	5	6
	Peter	**hat**	**kein**	**Tor**	**geschossen**	**.**	



Language Model

ARPA format

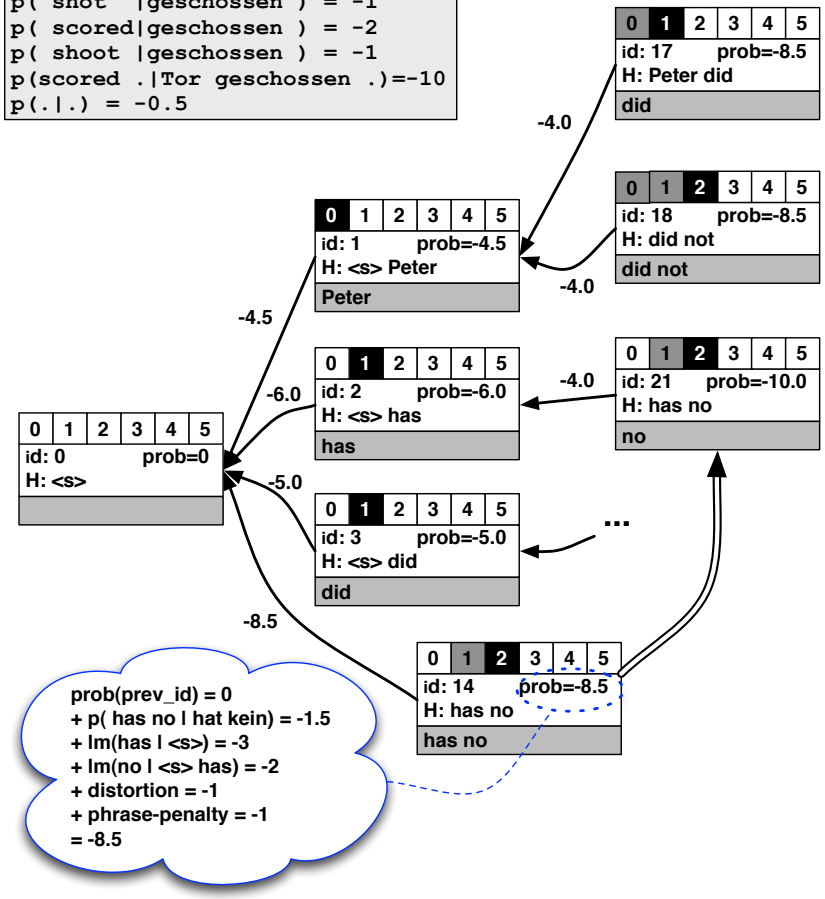
```
-0.5  .                -3
-4    goal             -0.5
-2    not              -1
-1    Peter
-6    scored           -3
-5    shoot            -0.1
-0.5  . </s>
-2    <s> Peter        -1
-3    <s> has          -0.5
-1    <s> did          -0.7
-0.5  did not          -0.5
-1.5  has no           -2
-2    shoot goals      -1
-1    <s> Peter did
-1    not shoot goals
-0.5  Peter did not
```

Recombination Example

Translation Model

```
p( Peter| Peter ) = -1.5
p( has  | hat   ) = -1
p( did  | hat   ) = -2
p( no   | kein  ) = -1
p( did not |hat kein)=-0.5
p( has no  |hat kein ) = -1.5
p( goal| Tor) = -2
p( goals|Tor ) = -3
p( shot  |geschossen ) = -1
p( scored|geschossen ) = -2
p( shoot |geschossen ) = -1
p(scored .|Tor geschossen .)=-10
p(.|.) = -0.5
```

	Peter	*has*	*not-a*	*goal*	*scored*	*.*	
-1	0	1	2	3	4	5	6
	Peter	**hat**	**kein**	**Tor**	**geschossen**	**.**	



Language Model

ARPA format

```
-0.5   .                -3
-4     goal             -0.5
-2     not              -1
-1     Peter
-6     scored           -3
-5     shoot            -0.1
-0.5   . </s>
-2     <s> Peter        -1
-3     <s> has          -0.5
-1     <s> did          -0.7
-0.5   did not          -0.5
-1.5   has no           -2
-2     shoot goals      -1
-1     <s> Peter did
-1     not shoot goals
-0.5   Peter did not
```

Recombination Example

Translation Model

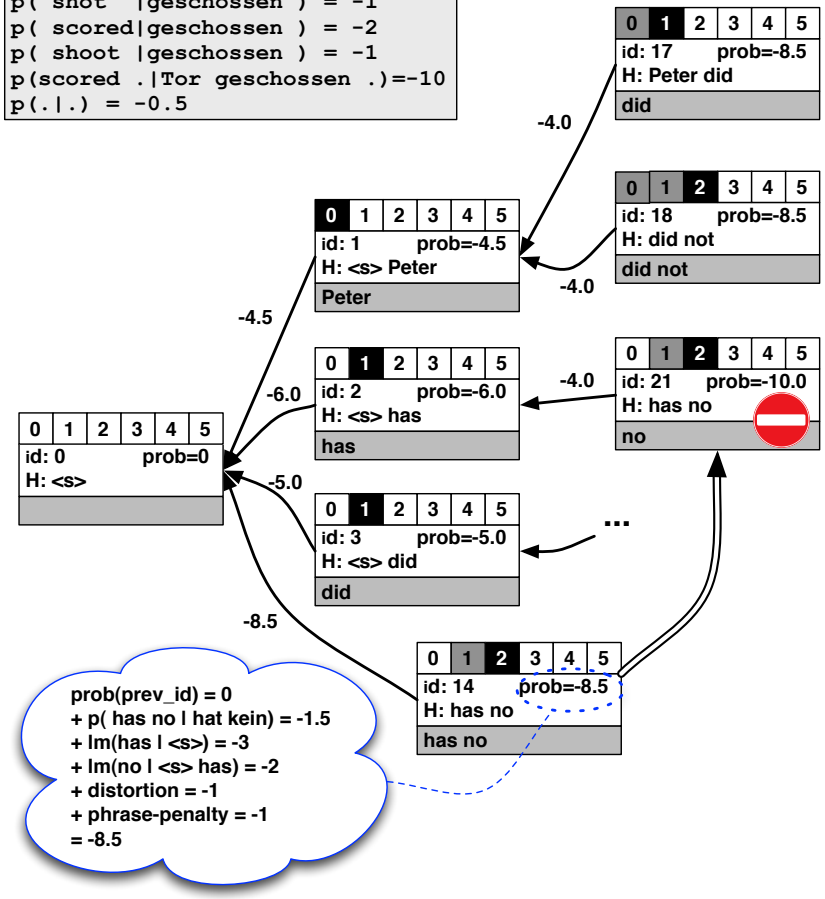
```
p( Peter| Peter ) = -1.5
p( has  | hat   ) = -1
p( did  | hat   ) = -2
p( no   | kein  ) = -1
p( did not |hat kein)=-0.5
p( has no  |hat kein ) = -1.5
p( goal| Tor) = -2
p( goals|Tor ) = -3
p( shot  |geschossen ) = -1
p( scored|geschossen ) = -2
p( shoot |geschossen ) = -1
p(scored .|Tor geschossen .)=-10
p(.|.) = -0.5
```

	Peter	*has*	*not-a*	*goal*	*scored*	*.*	
-1	0	1	2	3	4	5	6
	Peter	**hat**	**kein**	**Tor**	**geschossen**	**.**	

Language Model

ARPA format

```
-0.5  .                -3
-4    goal             -0.5
-2    not              -1
-1    Peter
-6    scored           -3
-5    shoot            -0.1
-0.5  . </s>
-2    <s> Peter        -1
-3    <s> has          -0.5
-1    <s> did          -0.7
-0.5  did not          -0.5
-1.5  has no           -2
-2    shoot goals      -1
-1    <s> Peter did
-1    not shoot goals
-0.5  Peter did not
```



Recombination Example

Translation Model

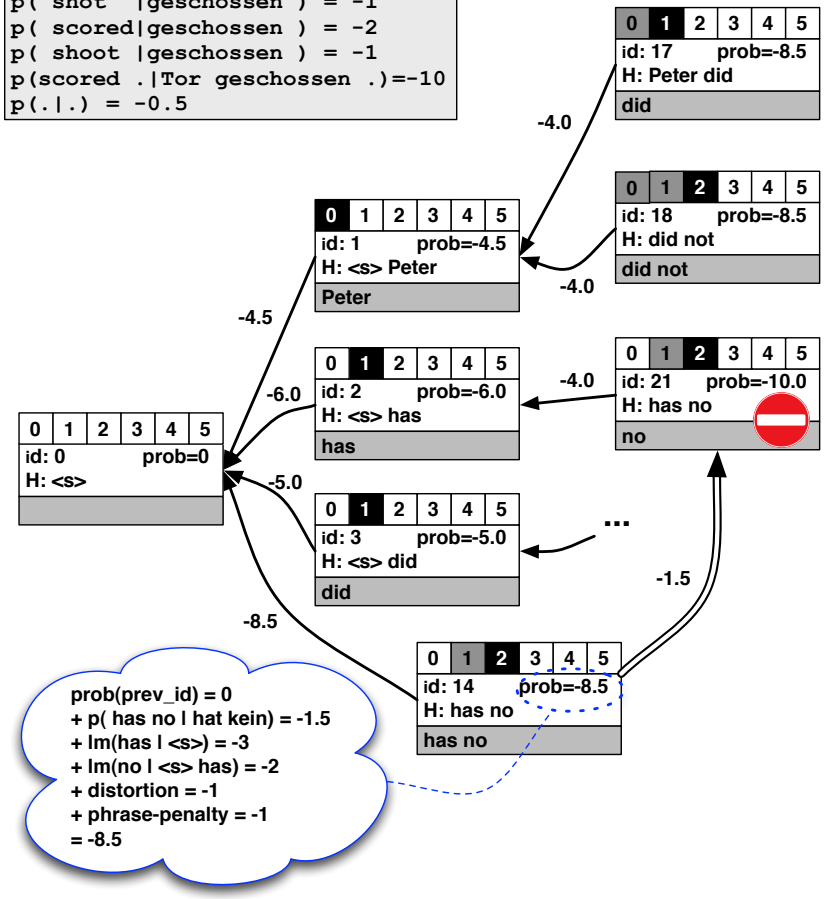
```
p( Peter| Peter ) = -1.5
p( has  | hat   ) = -1
p( did  | hat   ) = -2
p( no   | kein  ) = -1
p( did not |hat kein)=-0.5
p( has no  |hat kein ) = -1.5
p( goal| Tor) = -2
p( goals|Tor ) = -3
p( shot  |geschossen ) = -1
p( scored|geschossen ) = -2
p( shoot |geschossen ) = -1
p(scored .|Tor geschossen .)=-10
p(.|.) = -0.5
```

	Peter	*has*	*not-a*	*goal*	*scored*	*.*	
-1	0	1	2	3	4	5	6
	Peter	**hat**	**kein**	**Tor**	**geschossen**	**.**	

Language Model

ARPA format

```
-0.5  .                -3
-4    goal             -0.5
-2    not              -1
-1    Peter
-6    scored           -3
-5    shoot            -0.1
-0.5  . </s>
-2    <s> Peter        -1
-3    <s> has          -0.5
-1    <s> did          -0.7
-0.5  did not          -0.5
-1.5  has no           -2
-2    shoot goals      -1
-1    <s> Peter did
-1    not shoot goals
-0.5  Peter did not
```

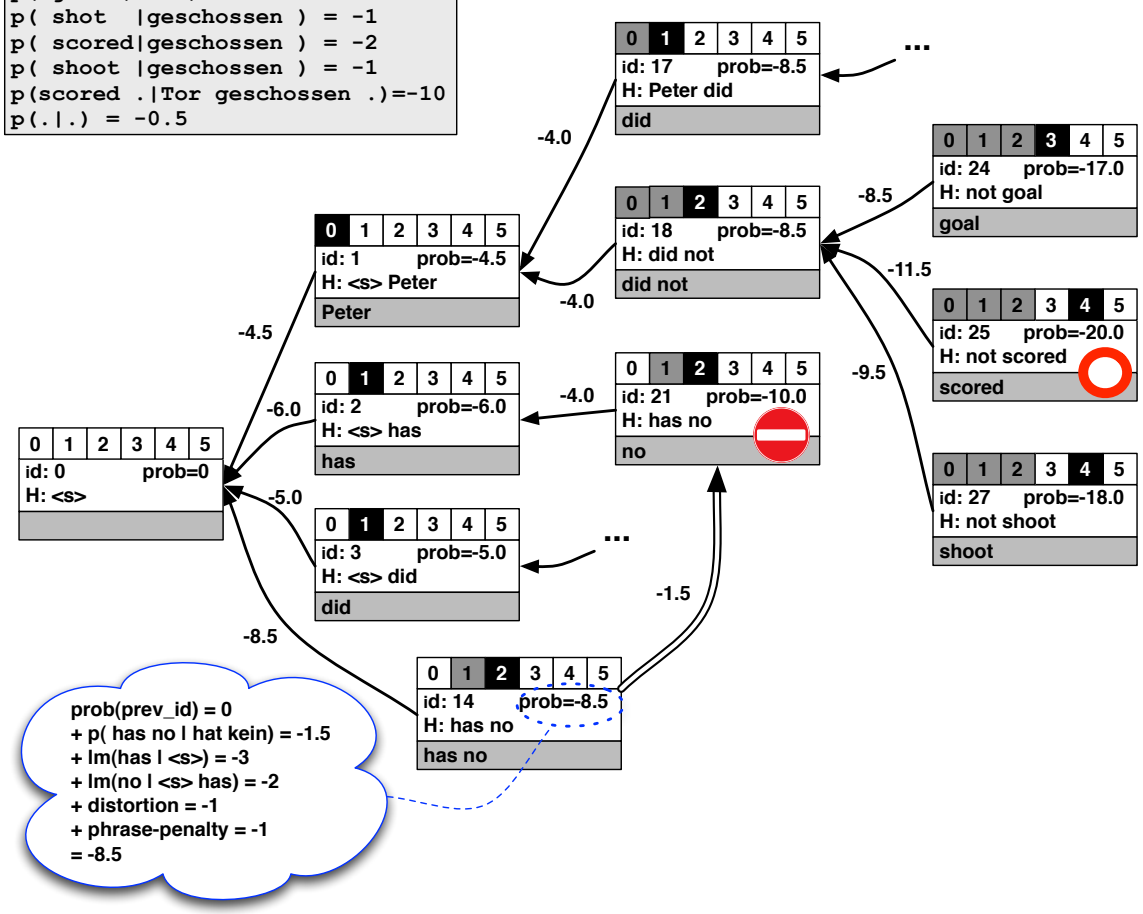


Recombination Example

Translation Model

```
p( Peter| Peter ) = -1.5
p( has  | hat   ) = -1
p( did  | hat   ) = -2
p( no   | kein  ) = -1
p( did not |hat kein)=-0.5
p( has no  |hat kein ) = -1.5
p( goal| Tor) = -2
p( goals|Tor ) = -3
p( shot  |geschossen ) = -1
p( scored|geschossen ) = -2
p( shoot |geschossen ) = -1
p(scored .|Tor geschossen .)=-10
p(.|.) = -0.5
```

	Peter	*has*	*not-a*	*goal*	*scored*	*.*	
-1	0	1	2	3	4	5	6
	Peter	**hat**	**kein**	**Tor**	**geschossen**	**.**	



Language Model

ARPA format

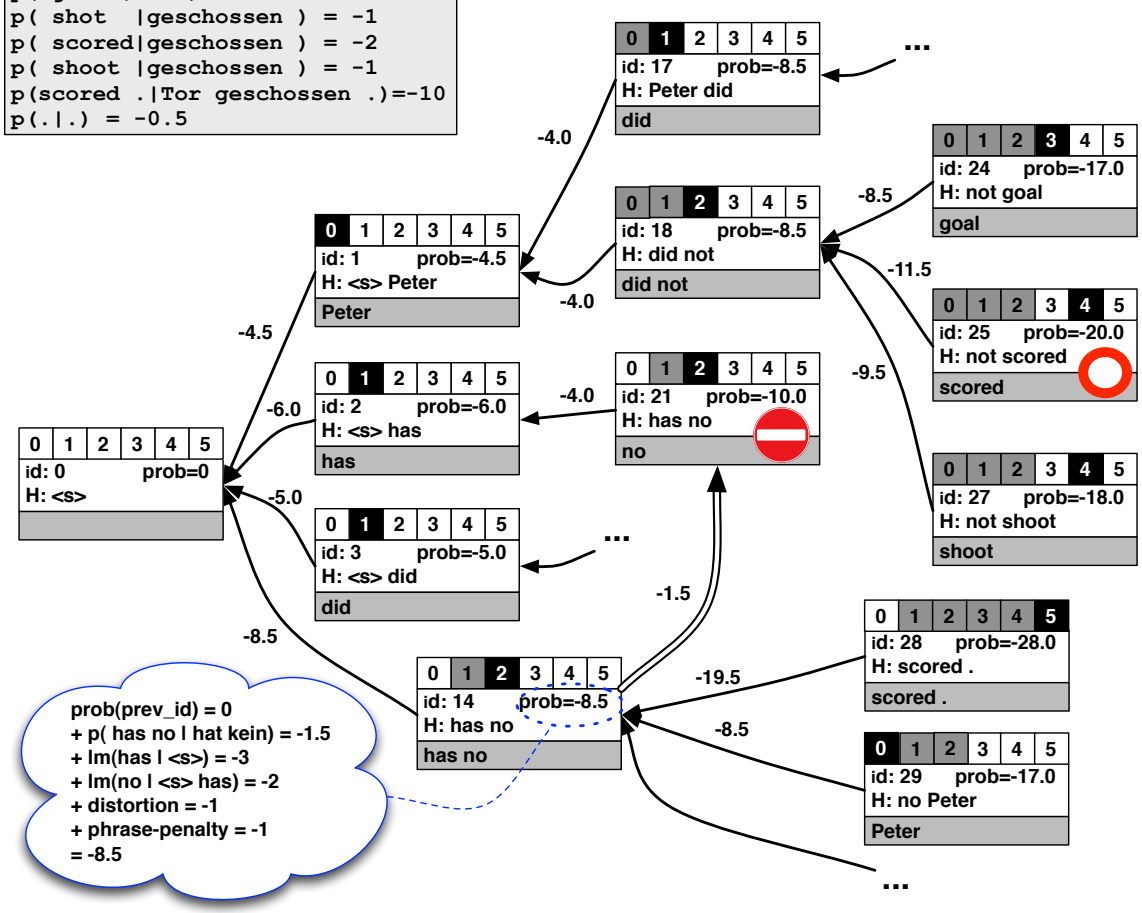
```
-0.5  .                -3
-4    goal             -0.5
-2    not              -1
-1    Peter
-6    scored           -3
-5    shoot            -0.1
-0.5  . </s>
-2    <s> Peter        -1
-3    <s> has          -0.5
-1    <s> did          -0.7
-0.5  did not          -0.5
-1.5  has no           -2
-2    shoot goals      -1
-1    <s> Peter did
-1    not shoot goals
-0.5  Peter did not
```

Recombination Example

Translation Model

```
p( Peter| Peter ) = -1.5
p( has  | hat   ) = -1
p( did  | hat   ) = -2
p( no   | kein  ) = -1
p( did not |hat kein)=-0.5
p( has no  |hat kein ) = -1.5
p( goal| Tor) = -2
p( goals|Tor ) = -3
p( shot  |geschossen ) = -1
p( scored|geschossen ) = -2
p( shoot |geschossen ) = -1
p(scored .|Tor geschossen .)=-10
p(.|.) = -0.5
```

	Peter	*has*	*not-a*	*goal*	*scored*	*.*	
-1	0	1	2	3	4	5	6
	Peter	**hat**	**kein**	**Tor**	**geschossen**	**.**	



Language Model

ARPA format

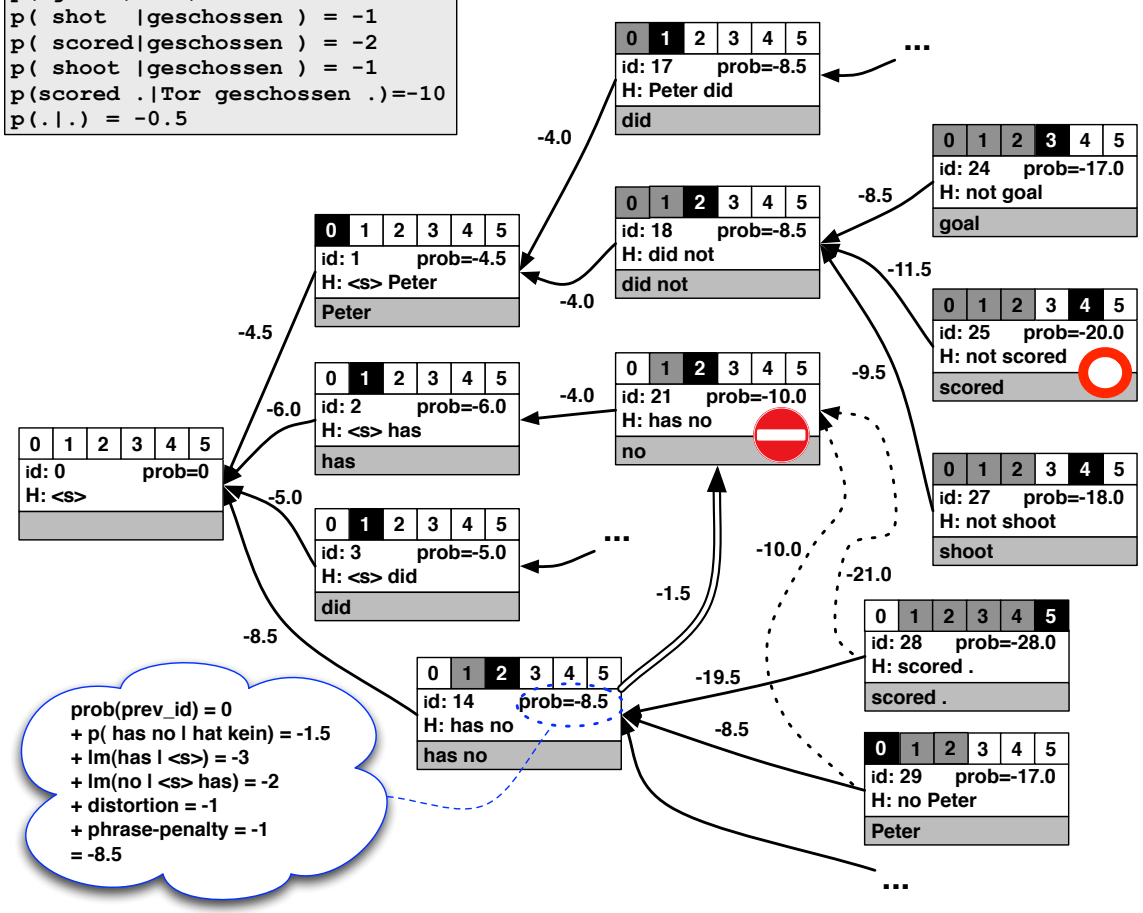
```
-0.5   .                -3
-4     goal             -0.5
-2     not              -1
-1     Peter
-6     scored           -3
-5     shoot            -0.1
-0.5   . </s>
-2     <s> Peter        -1
-3     <s> has          -0.5
-1     <s> did          -0.7
-0.5   did not          -0.5
-1.5   has no           -2
-2     shoot goals      -1
-1     <s> Peter did
-1     not shoot goals
-0.5   Peter did not
```

Recombination Example

Translation Model

```
p( Peter| Peter ) = -1.5
p( has  | hat  ) = -1
p( did  | hat  ) = -2
p( no   | kein ) = -1
p( did not |hat kein)=-0.5
p( has no  |hat kein ) = -1.5
p( goal| Tor) = -2
p( goals|Tor ) = -3
p( shot  |geschossen ) = -1
p( scored|geschossen ) = -2
p( shoot |geschossen ) = -1
p(scored .|Tor geschossen .)=-10
p(.|.) = -0.5
```

	Peter	*has*	*not-a*	*goal*	*scored*	*.*	
-1	0	1	2	3	4	5	6
	Peter	**hat**	**kein**	**Tor**	**geschossen**	**.**	



Language Model

ARPA format

```
-0.5   .                -3
-4     goal             -0.5
-2     not              -1
-1     Peter
-6     scored           -3
-5     shoot            -0.1
-0.5   . </s>
-2     <s> Peter        -1
-3     <s> has          -0.5
-1     <s> did          -0.7
-0.5   did not          -0.5
-1.5   has no           -2
-2     shoot goals      -1
-1     <s> Peter did
-1     not shoot goals
-0.5   Peter did not
```

Recombination Example

Translation Model

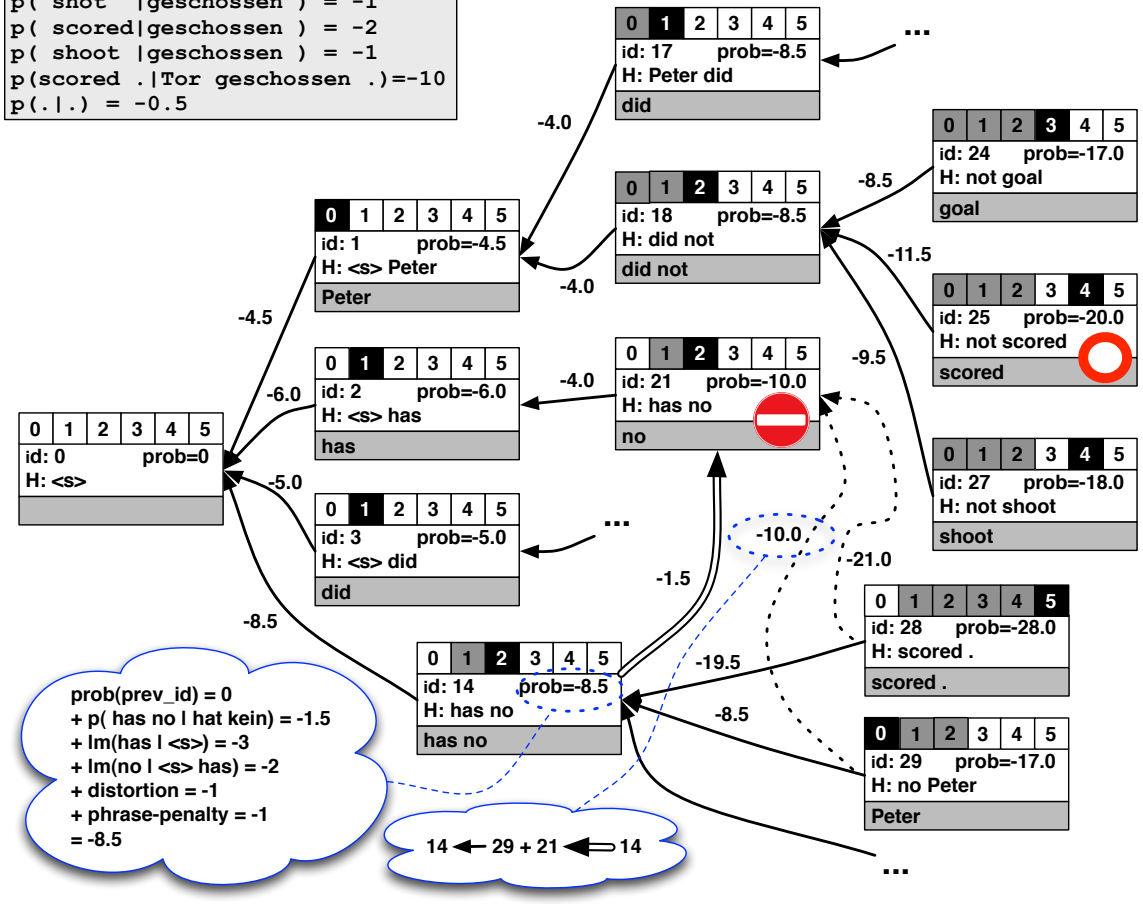
```
p( Peter| Peter ) = -1.5
p( has  | hat   ) = -1
p( did  | hat   ) = -2
p( no   | kein  ) = -1
p( did not |hat kein)=-0.5
p( has no  |hat kein ) = -1.5
p( goal| Tor) = -2
p( goals|Tor ) = -3
p( shot  |geschossen ) = -1
p( scored|geschossen ) = -2
p( shoot |geschossen ) = -1
p(scored .|Tor geschossen .)=-10
p(.|.) = -0.5
```

	Peter	*has*	*not-a*	*goal*	*scored*	*.*	
-1	0	1	2	3	4	5	6
	Peter	**hat**	**kein**	**Tor**	**geschossen**	**.**	

Language Model

ARPA format

```
-0.5  .                -3
-4    goal             -0.5
-2    not              -1
-1    Peter
-6    scored           -3
-5    shoot            -0.1
-0.5  . </s>
-2    <s> Peter        -1
-3    <s> has          -0.5
-1    <s> did          -0.7
-0.5  did not          -0.5
-1.5  has no           -2
-2    shoot goals      -1
-1    <s> Peter did
-1    not shoot goals
-0.5  Peter did not
```



Search Space Pruning

- DP decoding and recombination reduces the complexity from $O(n!)$ to $O(n^2 2^n)$, better, but still exponential
- DP decoding is safe in the sense that it does not remove partial hypothesis during search that could have turned out to be optimal
  - I.e., it does not make search errors (we come back to this later)
- For efficient or tractable SMT we have to resort to more aggressive methods: pruning
  - Heuristic pruning methods
  - Beam search pruning
- Pruning is not guaranteed to avoid making search errors!

Heuristic Search Space Constraints

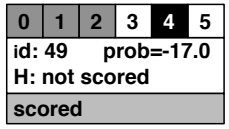
- Things that don't even enter the generation process of $\tilde{E}$
- During the generation of $\tilde{E}$ from E^* we can decide to ignore certain hypotheses
  - These decisions do not depend on the model parameters
  - Therefore, they can not be optimized for and are heuristically set as meta-parameters
- These search space restrictions include:
  - **Distortion limit:** Do no consider hypotheses with a distortion cost greater than the limit
  - **Translation limit:** Only consider the top n translations of each foreign phrase (typically $n \in \{20, 30, 50\}$)
  - **Minimum translation probability:** Only consider phrase translations where $p(\bar{e}|\bar{f}) \geq \theta$ and $p(\bar{f}|\bar{e}) \geq \theta$

Beam Search Pruning

- There are two types of beam search pruning
  - **Histogram pruning:** fixed maximum number of states per stack (we come back to this after having introduced multi-stacks)
  - **Threshold pruning:** only keep states whose probability is θ times the probability of the best *comparable* state, where $0 < \theta < 1$
- When are two states comparable?
  - If they solve the same subproblem (see DP decoding)
    - But this does not allow for more aggressive pruning beyond DP decoding
  - If they have translated the same number of foreign words
    - Cover vectors do not have to be identical
    - Language model histories can be different
    - Last translated foreign word position can differ

Beam Search Threshold Pruning

- The following two states are comparable
  - Assuming a beam width of 0.15, state `id:42` could be pruned



Translation Model

```
...
p( goal| Tor) = -3
p( goals|Tor ) = -3
p( shot  |geschossen ) = -1
p( scored|geschossen ) = -2
p( shoot |geschossen ) = -1
...
p(.|.) = -0.5
```

- Assume that we proceeded:
  - `id:42`: word 4 ('geschossen') could be translated with probability -1 and no additional distortion cost $\rightarrow$ `prob=-21`
  - `id:49`: word 3 ('Tor) could be translated with probability -3 and distortion cost -2 $\rightarrow$ `prob=-22`
  - Problem: Now `id:42` (the one we wanted to prune) leads to a higher probability than `id:49` (the one we wanted to keep) $\rightarrow$ search error!

Future Costs

- When pruning we should also two types of costs:
  - **Current costs:** the costs of a state given the words that have been translated so far
  - **Future costs:** the *minimal* costs that we are to incur when expanding a state to cover all words (also known as *rest costs*)
- Exact computation of the rest costs is as expensive as actual decoding
  - In order to prune states we first have to do a full search (pointless!)
- Future costs are estimated
  - For each span of foreign tokens $[i,j]$
  - Based on translation model estimate
  - Distortion model estimate
  - Language model estimate
- When comparing states we consider the sum of current and future costs of each state

Future Cost Computation

- A future cost estimate can be computed using dynamic programming
- Initialization: For each span $[i,j] = \bar{f}$
  - $F[i,j] = \max_{\bar{e}} \log(p_{TM}(\bar{e}|\bar{f}) \cdot p_{LM}(\bar{e}))$, if $n_{1+}(\bullet|\bar{f}) > 0$
  - $F[i,j] = -10 + \log(p_{LM}(\bar{f}))$, else, and if $i = j$
  - $F[i,j] = -10000$, else
- Iteration: For $l = 2, \ldots, n$
  - For each span $[i,j]$, s.t. $j - i + 1 = l$
    - $F[i,j] = \max\{\max_{i \leq k < j} F[i,k] + F[k+1,j],\ F[i,j]\}$
- Can be done in $O(n^2) \ll O(2^n)$
- Note, language model costs are *estimated* in isolation
  - Estimation of language costs in context requires considering all possible reorderings $\rightarrow O(2^n)$
  - Future costs for `id:49` $= F[3,3] + F[5,5]$
- What about future cost estimation of linear distortion?

A* Search

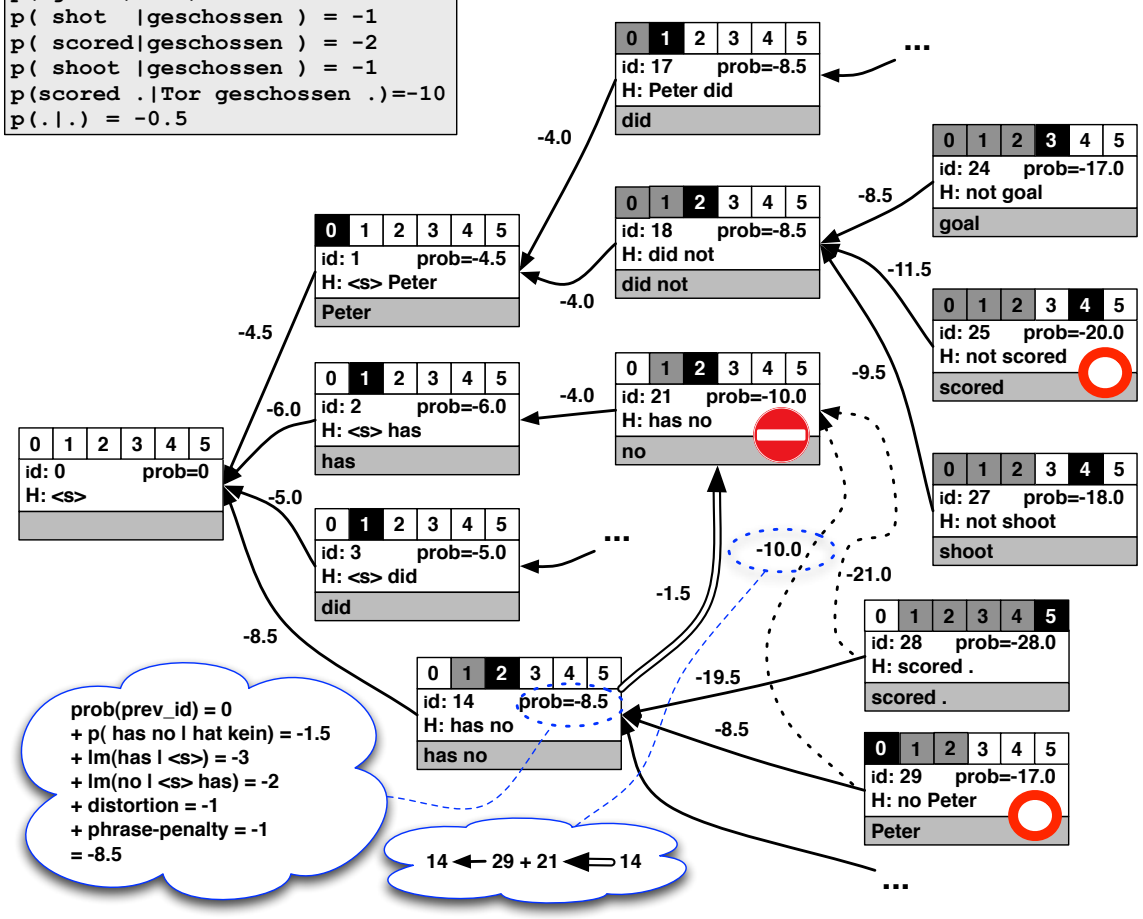
- Future cost or rest cost estimation is closely related to A* search
  - Classic search algorithm in Artificial Intelligence by Hart et al. (1968)
- If $h(s)$ is a heuristic function to compute the future costs for state s, e.g., $s = \texttt{id:49}$
  - h is *admissible* if it never overestimates the cost of reaching a goal state from s
    Here, h is *admissible* if it never underestimates the probability of reaching the final state from s
  - Underestimation of the future cost (i.e. probability) can lead to a state being erroneously pruned

Search Strategy

Translation Model

```
p( Peter| Peter ) = -1.5
p( has  | hat  ) = -1
p( did  | hat  ) = -2
p( no   | kein ) = -1
p( did not |hat kein)=-0.5
p( has no  |hat kein ) = -1.5
p( goal| Tor) = -2
p( goals|Tor ) = -3
p( shot  |geschossen ) = -1
p( scored|geschossen ) = -2
p( shoot |geschossen ) = -1
p(scored .|Tor geschossen .)=-10
p(.|.) = -0.5
```

	Peter	*has*	*not-a*	*goal*	*scored*	*.*	
-1	0	1	2	3	4	5	6
	Peter	**hat**	**kein**	**Tor**	**geschossen**	**.**	



Language Model

ARPA format

```
-0.5   .                -3
-4     goal             -0.5
-2     not              -1
-1     Peter
-6     scored           -3
-5     shoot            -0.1
-0.5   . </s>
-2     <s> Peter        -1
-3     <s> has          -0.5
-1     <s> did          -0.7
-0.5   did not          -0.5
-1.5   has no           -2
-2     shoot goals      -1
-1     <s> Peter did
-1     not shoot goals
-0.5   Peter did not
```

Search Strategy

Translation Model

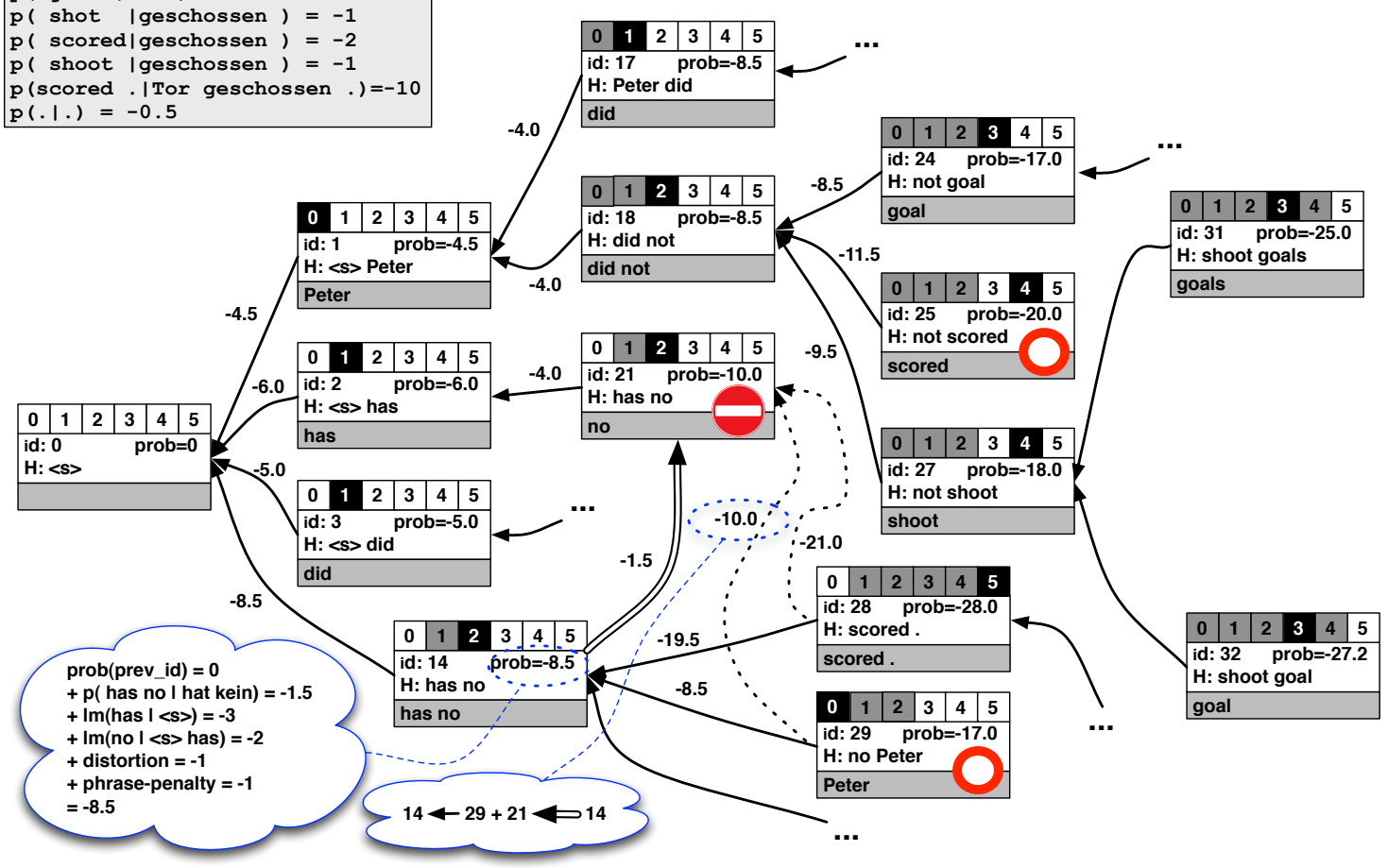
```
p( Peter| Peter ) = -1.5
p( has  | hat  ) = -1
p( did  | hat  ) = -2
p( no   | kein ) = -1
p( did not |hat kein)=-0.5
p( has no  |hat kein ) = -1.5
p( goal| Tor) = -2
p( goals|Tor ) = -3
p( shot  |geschossen ) = -1
p( scored|geschossen ) = -2
p( shoot |geschossen ) = -1
p(scored .|Tor geschossen .)=-10
p(.|.) = -0.5
```

	Peter	*has*	*not-a*	*goal*	*scored*	*.*	
-1	0	1	2	3	4	5	6
	Peter	**hat**	**kein**	**Tor**	**geschossen**	**.**	

Language Model

ARPA format

```
-0.5  .                -3
-4    goal             -0.5
-2    not              -1
-1    Peter
-6    scored           -3
-5    shoot            -0.1
-0.5  . </s>
-2    <s> Peter        -1
-3    <s> has          -0.5
-1    <s> did          -0.7
-0.5  did not          -0.5
-1.5  has no           -2
-2    shoot goals      -1
-1    <s> Peter did
-1    not shoot goals
-0.5  Peter did not
```



Search Strategy

Translation Model

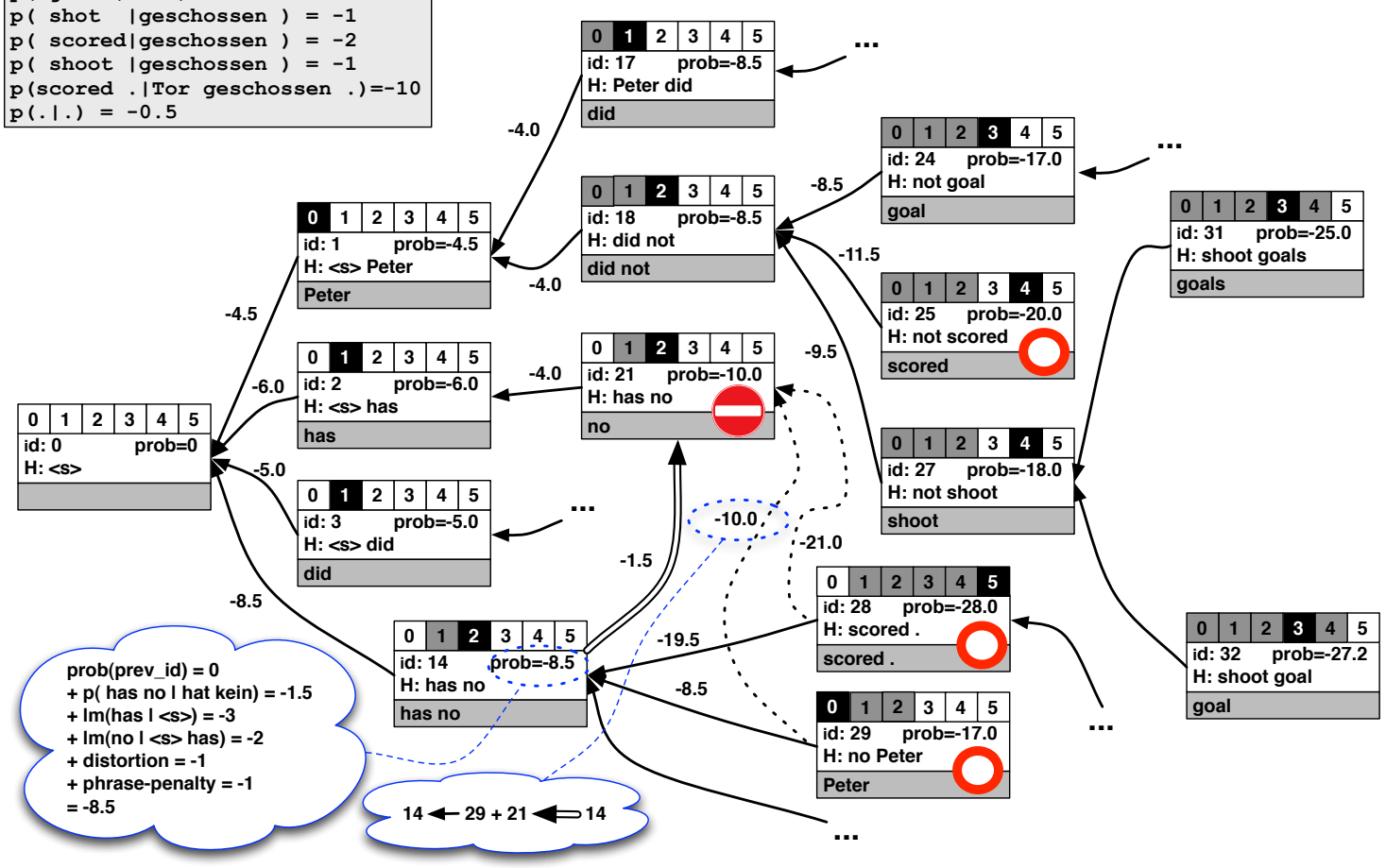
```
p( Peter| Peter ) = -1.5
p( has  | hat  ) = -1
p( did  | hat  ) = -2
p( no   | kein ) = -1
p( did not |hat kein)=-0.5
p( has no  |hat kein ) = -1.5
p( goal| Tor) = -2
p( goals|Tor ) = -3
p( shot  |geschossen ) = -1
p( scored|geschossen ) = -2
p( shoot |geschossen ) = -1
p(scored .|Tor geschossen .)=-10
p(.|.) = -0.5
```

	Peter	*has*	*not-a*	*goal*	*scored*	*.*	
-1	0	1	2	3	4	5	6
	Peter	**hat**	**kein**	**Tor**	**geschossen**	**.**	

Language Model

ARPA format

```
-0.5  .                -3
-4    goal             -0.5
-2    not              -1
-1    Peter
-6    scored           -3
-5    shoot            -0.1
-0.5  . </s>
-2    <s> Peter        -1
-3    <s> has          -0.5
-1    <s> did          -0.7
-0.5  did not          -0.5
-1.5  has no           -2
-2    shoot goals      -1
-1    <s> Peter did
-1    not shoot goals
-0.5  Peter did not
```



Search Strategy

Translation Model

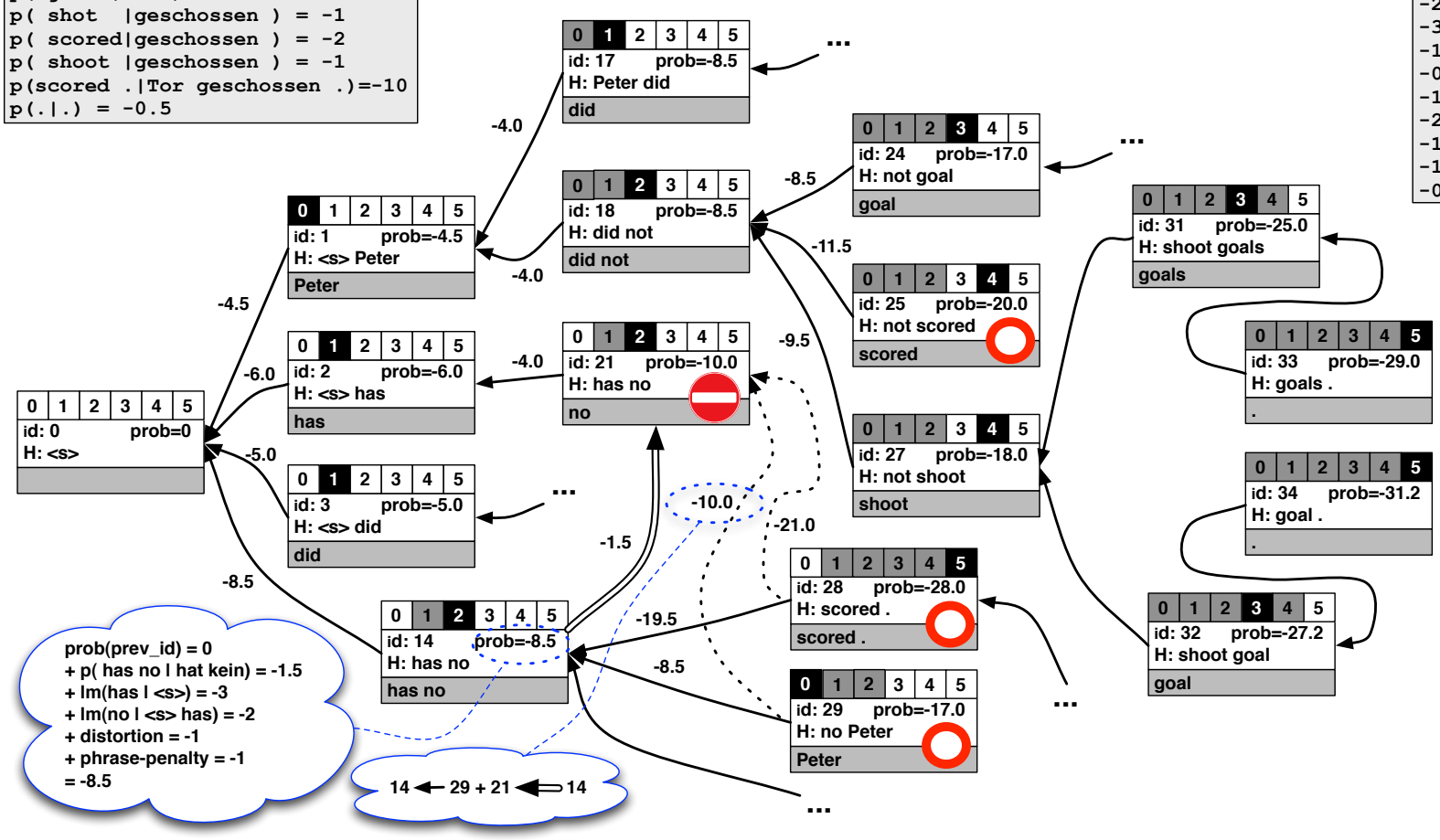
```
p( Peter| Peter ) = -1.5
p( has  | hat  ) = -1
p( did  | hat  ) = -2
p( no   | kein ) = -1
p( did not |hat kein)=-0.5
p( has no  |hat kein ) = -1.5
p( goal| Tor) = -2
p( goals|Tor ) = -3
p( shot  |geschossen ) = -1
p( scored|geschossen ) = -2
p( shoot |geschossen ) = -1
p(scored .|Tor geschossen .)=-10
p(.|.) = -0.5
```

	Peter	has	not-a	goal	scored	.	
-1	0	1	2	3	4	5	6
	Peter	hat	kein	Tor	geschossen	.	

Language Model

ARPA format

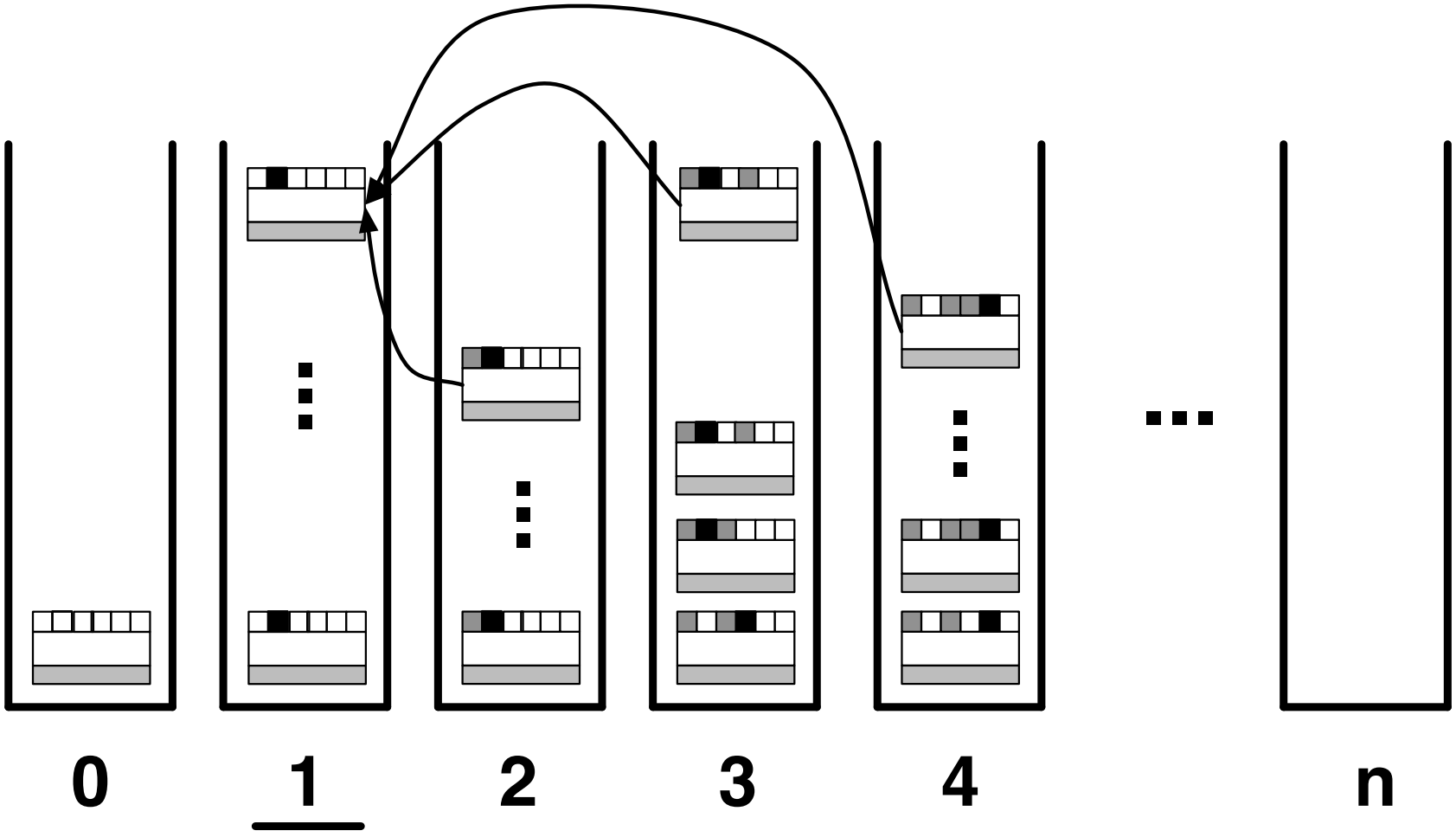
```
-0.5  .                -3
-4    goal             -0.5
-2    not              -1
-1    Peter
-6    scored           -3
-5    shoot            -0.1
-0.5  . </s>
-2    <s> Peter        -1
-3    <s> has          -0.5
-1    <s> did          -0.7
-0.5  did not          -0.5
-1.5  has no           -2
-2    shoot goals      -1
-1    <s> Peter did
-1    not shoot goals
-0.5  Peter did not
```



Search Strategy

- In general, two types of search strategies can be distinguished
  - Depth-first search: Expand hypothesis along one path until final state has been reach, the explore the deepest alternative path
  - Breadth-first search: Expand all hypotheses of depth d (creating multiple paths) before expanding hypotheses of depth $d+1$
- In phrase-based decoding the breadth-first balancing criterion (depth) is the number of foreign words covered
  - Expand all hypotheses that cover m words before expanding hypotheses covering $m+1$ words
  - All hypotheses (states) covering m words are put in the same stack

Multi-Stack Decoding



Multi-Stack Decoding

- All hypotheses (states) that are comparable for pruning are in the same stack
- All hypotheses that can be recombined are in the same stack
- Before expanding the hypotheses in stack m
  - All hypotheses (s) in stack m are sorted according to $\text{current\_cost}(s) + \text{future\_cost}(s)$
  - Prune hypotheses that are outside of the beam of the top-scoring hypothesis
  - Recombine all remaining hypotheses (where possible)
  - **Histogram pruning:** If there are k hypotheses in stack m and $k > l$, where l is the histogram pruning limit remove the $k - l$ lowest scoring hypotheses

Derivation

- Given highest scoring hypothesis (s_m^*) in stack n (where n is the length of the foreign sentence) we can extract the Viterbi translation:
  - $(s_0, \ldots, \text{back\_pointer}(\text{back\_pointer}(s_m^*)), \text{back\_pointer}(s_m^*), s_m^*)$
- Actually, this gives us the best *derivation*, from which we can extract the best surface translation
- Multiple derivations can lead to the same translation!
- A derivation captures which foreign phrases have been translated into which English phrases, and in which order

  $[_0\text{Peter}]_0\ [_1\text{has not}]_2\ [_4\text{scored}]_4\ [_3\text{goals}]_3\ [_5\text{.}]_5$

  The positions of foreign phrases can be recovered from the coverage vectors if represented as bit-vectors:

  $$\text{cov}(s_m) - \text{cov}(\text{back\_pointer}(s_m))$$

Viterbi Approximation

- Our Viterbi translation is actually the Viterbi derivation and its corresponding translation
- This lead to the Viterbi approximation:

  $$\text{trans}(f) = \text{yield}(\arg\max_{(e,d)} p(f,d|e) \cdot p(e))$$

  where d is a derivation and $\text{yield}((e,d)) = e$
- Different derivations with the same yield are in competition, while they *should* reinforce each other
- In exact decoding we sum over all derivations:

  $$\text{trans}(f) = \arg\max_e \sum_d p(f,d|e) \cdot p(e))$$
- Unfortunately, exact decoding is computationally intractable

N-best List Extraction

- In most cases the 1-best translation is sufficient
- Cases where n-best lists are desirable include
  - Parameter tuning
  - N-best list rescoring
  - Presentation of translation alternatives to user
- Given the search lattice L we can extract the n-best paths by using standard algorithms
  - Eppstein's k shortest paths algorithm (SIAM J. Computing, 1999)
  - Yen's k shortest paths algorithm (Management Science, 1971)

N-best List Example

- Simple example (10-best)

Rank	Translation	Probability
1.	the well - known constants .	−2.54
2.	the principles are known .	−2.62
3.	the known constants .	−2.78
4.	the principles are well known .	−2.80
5.	the principles known .	−2.81
6.	it is a well - known constants .	−2.84
7.	the principles known .	−2.85
8.	the well - known principles .	−2.90
9.	the well - known constants ,	−2.91
10.	, the well - known constants .	−2.95

Minimum Bayes Risk

- While exact decoding is computationally intractable, we can approximate it by focusing on n-best lists
- This is known as Minimum Bayes Risk (MBR) computation
- Let $\hat{E}$ be the multi-set of n-best translations for a foreign sentence f
  - For each $e \in \hat{E}$: $p_{MBR}(e|f) = \sum_{e' \in \hat{E}} S(e,e') \cdot p_M(e')$

    where $p_M(e)$ is the Viterbi approximated probability of translation f into e (in probability space *not* log space!)

    $S(e,e')$ is a similarity function which can be defined as

    Zero-One simliarity: $S(e,e') = 1$ if $e = e'$, else $S(e,e') = 0$
    More flexible overlap criterion such as $\text{BLEU}(e,e')$
- MBR boosts translation candidates that are in consensus with many other (high scoring) candidates

Rescoring

- Some features are difficult to fully integrate into a DP decoder
  - Typically these are features that apply to whole sentences
- Examples include
  - Discriminative LMs with parse or whole sentence features (see earlier lecture on LMs for SMT)
  - Sentence length features
  - Higher-order n-gram language models
- Translation candidates in an n-best can be re-scored and re-ranked based on those feature values
  - In combination with features used during DP decoding
  - The highest re-scored candidate is returned as translation
- Rescoring is
  - Fairly easy to realize
  - Limited effectiveness as it depends on $\hat{E}$ only and does not affect $\tilde{E}$

Search Errors

- How correct is a decoder?
  - The correctness of a decoder is measured as its search error
- Search errors occur when the search algorithm misses the highest scoring translation and produces a lower scoring translation instead
- Determining the highest scoring possible translation is not possible in general
  - Instead this is approximated by using very conservative pruning settings (very slow!) and then compare it to using 'normal' pruning settings
- Causes of search errors:
  - Poor future cost estimation
  - Too aggressive pruning

Model Errors

- How good are our models (in combination with the decoder)?
  - Measured as the model error
- Needs an objective function to measure MT quality
  - BLEU
  - TER
- Model errors occur if the highest scoring translation is of lower quality than a lower scoring translation
- Causes of model errors:
  - Noise/errors in the translation model
  - Noise/errors in the language model

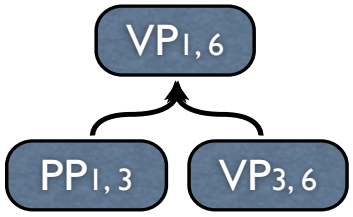
Efficiency Improvements

- Since decoding is a complex task, speed performance can be an issue
  - Use more aggressive pruning $\rightarrow$ search errors
  - Caching of repeated function calls (such as language model calls)
  - Parallelization: Most systems translate sentences independent of each other $\rightarrow$ sentences in a document can be translated in parallel
  - Delaying and prioritization of expensive function calls (e.g., language model calls)
- The right balance between speed and quality depends on the end-user task
  - Quick and dirty for relevance assessment (exploration,triage)
  - Medium quality for further processing (data mining, information extraction)
  - High quality for dissemination

Cube Pruning

- Cube pruning (Huang and Chiang, 2007) addresses the issue of high cost language model computations
- Computing the language model cost for a hypothesis continuation
  - is delayed
  - limited to the k most promising continuations
- Originally developed for hierarchical decoding but can be adapted to phrase-based SMT
- More sensitive to search errors
  - Compensated by efficiency improvements allowing for less aggressive pruning
- Following slides taken from Huang and Chiang (2007)

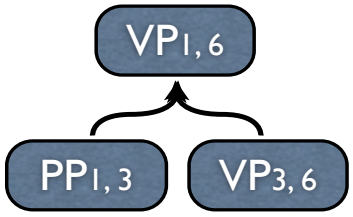
Cube Pruning



monotonic grid?

		$(\text{PP}_{1,3}^{\text{with} \star \text{Sharon}})$	$(\text{PP}_{1,3}^{\text{along} \star \text{Sharon}})$	$(\text{PP}_{1,3}^{\text{with} \star \text{Shalong}})$
		1.0	3.0	8.0
$(\text{VP}_{3,6}^{\text{held} \star \text{meeting}})$	1.0	2.0	4.0	9.0
$(\text{VP}_{3,6}^{\text{held} \star \text{talk}})$	1.1	2.1	4.1	9.1
$(\text{VP}_{3,6}^{\text{hold} \star \text{conference}})$	3.5	4.5	6.5	11.5

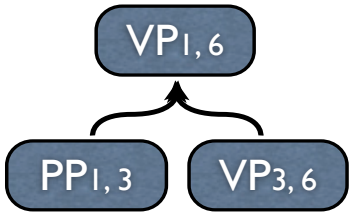
Cube Pruning



non-monotonic grid
due to LM combo costs

	$(\text{PP}_{1,3}^{\text{with} \star \text{Sharon}})$ 1.0	$(\text{PP}_{1,3}^{\text{along} \star \text{Sharon}})$ 3.0	$(\text{PP}_{1,3}^{\text{with} \star \text{Shalong}})$ 8.0
$(\text{VP}_{3,6}^{\text{held} \star \text{meeting}})$ 1.0	2.0 + 0.5	4.0 + 5.0	9.0 + 0.5
$(\text{VP}_{3,6}^{\text{held} \star \text{talk}})$ 1.1	2.1 + 0.3	4.1 + 5.4	9.1 + 0.3
$(\text{VP}_{3,6}^{\text{hold} \star \text{conference}})$ 3.5	4.5 + 0.6	6.5 + 10.5	11.5 + 0.6

Cube Pruning



bigram (meeting, with)

non-monotonic grid
due to LM combo costs

$(\text{VP}_{3,6}^{\text{held} \star \text{meeting}})$

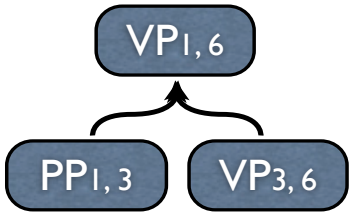
$(\text{VP}_{3,6}^{\text{held} \star \text{talk}})$

$(\text{VP}_{3,6}^{\text{hold} \star \text{conference}})$

$(\text{PP}_{1,3}^{\text{with} \star \text{Sharon}})$ $(\text{PP}_{1,3}^{\text{along} \star \text{Sharon}})$ $(\text{PP}_{1,3}^{\text{with} \star \text{Shalong}})$

	1.0	3.0	8.0
1.0	2.0 + 0.5	4.0 + 5.0	9.0 + 0.5
1.1	2.1 + 0.3	4.1 + 5.4	9.1 + 0.3
3.5	4.5 + 0.6	6.5 + 10.5	11.5 + 0.6

Cube Pruning



non-monotonic grid
due to LM combo costs

	$(\text{PP}_{1,3}^{\text{with} \star \text{Sharon}})$ 1.0	$(\text{PP}_{1,3}^{\text{along} \star \text{Sharon}})$ 3.0	$(\text{PP}_{1,3}^{\text{with} \star \text{Shalong}})$ 8.0
$(\text{VP}_{3,6}^{\text{held} \star \text{meeting}})$ 1.0	2.5	9.0	9.5
$(\text{VP}_{3,6}^{\text{held} \star \text{talk}})$ 1.1	2.4	9.5	9.4
$(\text{VP}_{3,6}^{\text{hold} \star \text{conference}})$ 3.5	5.1	17.0	12.1

Cube Pruning

k-best parsing
(Huang and Chiang, 2005)

- a priority queue of candidates
- extract the best candidate

		$(\text{PP}_{1,3}^{\text{with} \star \text{Sharon}})$ 1.0	$(\text{PP}_{1,3}^{\text{along} \star \text{Sharon}})$ 3.0	$(\text{PP}_{1,3}^{\text{with} \star \text{Shalong}})$ 8.0
$(\text{VP}_{3,6}^{\text{held} \star \text{meeting}})$	1.0	2.5	9.0	9.5
$(\text{VP}_{3,6}^{\text{held} \star \text{talk}})$	1.1	2.4	9.5	9.4
$(\text{VP}_{3,6}^{\text{hold} \star \text{conference}})$	3.5	5.1	17.0	12.1

Cube Pruning

k-best parsing
(Huang and Chiang, 2005)

- a priority queue of candidates
- extract the best candidate
- push the two successors

		$(\text{PP}_{1,3}^{\text{with} \star \text{Sharon}})$	$(\text{PP}_{1,3}^{\text{along} \star \text{Sharon}})$	$(\text{PP}_{1,3}^{\text{with} \star \text{Shalong}})$
		1.0	3.0	8.0
$(\text{VP}_{3,6}^{\text{held} \star \text{meeting}})$	1.0	2.5	9.0	9.5
$(\text{VP}_{3,6}^{\text{held} \star \text{talk}})$	1.1	2.4	9.5	9.4
$(\text{VP}_{3,6}^{\text{hold} \star \text{conference}})$	3.5	5.1	17.0	12.1

Cube Pruning

k-best parsing
(Huang and Chiang, 2005)

- a priority queue of candidates
- extract the best candidate
- push the two successors

	$(\text{PP}_{1,3}^{\text{with} \star \text{Sharon}})$	$(\text{PP}_{1,3}^{\text{along} \star \text{Sharon}})$	$(\text{PP}_{1,3}^{\text{with} \star \text{Shalong}})$
	1.0	3.0	8.0
$(\text{VP}_{3,6}^{\text{held} \star \text{meeting}})$ 1.0	2.5	9.0	9.5
$(\text{VP}_{3,6}^{\text{held} \star \text{talk}})$ 1.1	2.4	9.5	9.4
$(\text{VP}_{3,6}^{\text{hold} \star \text{conference}})$ 3.5	5.1	17.0	12.1

Cube Pruning

items are popped out-of-order

solution: keep a buffer of pop-ups

2.5 2.4 5.1

		$(\text{PP}_{1,3}^{\text{with} \star \text{Sharon}})$ 1.0	$(\text{PP}_{1,3}^{\text{along} \star \text{Sharon}})$ 3.0	$(\text{PP}_{1,3}^{\text{with} \star \text{Shalong}})$ 8.0
$(\text{VP}_{3,6}^{\text{held} \star \text{meeting}})$	1.0	2.5	9.0	9.5
$(\text{VP}_{3,6}^{\text{held} \star \text{talk}})$	1.1	2.4	9.5	9.4
$(\text{VP}_{3,6}^{\text{hold} \star \text{conference}})$	3.5	5.1	17.0	12.1

Cube Pruning

items are popped out-of-order
solution: keep a buffer of pop-ups

2.5 2.4 5.1

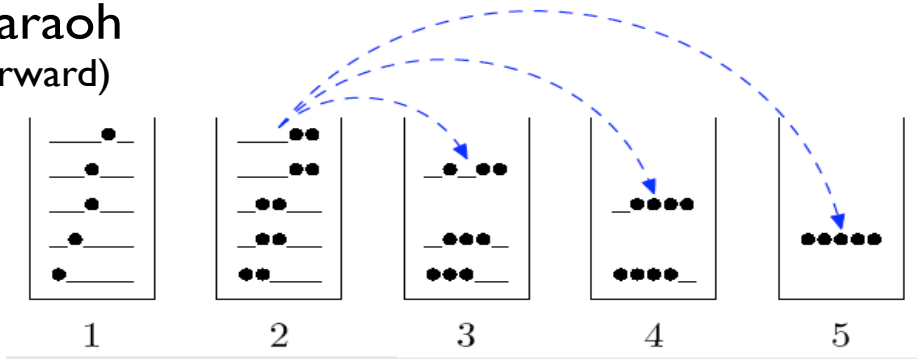
finally re-sort the buffer
and return inorder:

2.4 2.5 5.1

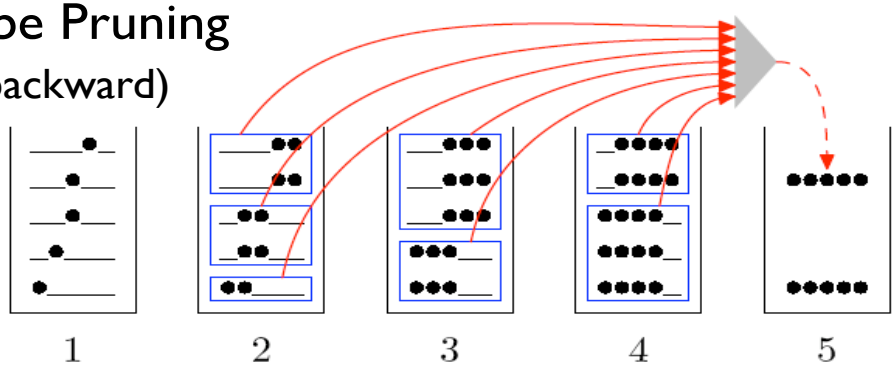
	$(\text{PP}_{1,3}^{\text{with} \star \text{Sharon}})$ 1.0	$(\text{PP}_{1,3}^{\text{along} \star \text{Sharon}})$ 3.0	$(\text{PP}_{1,3}^{\text{with} \star \text{Shalong}})$ 8.0
$(\text{VP}_{3,6}^{\text{held} \star \text{meeting}})$ 1.0	2.5	9.0	9.5
$(\text{VP}_{3,6}^{\text{held} \star \text{talk}})$ 1.1	2.4	9.5	9.4
$(\text{VP}_{3,6}^{\text{hold} \star \text{conference}})$ 3.5	5.1	17.0	12.1

Cube Pruning

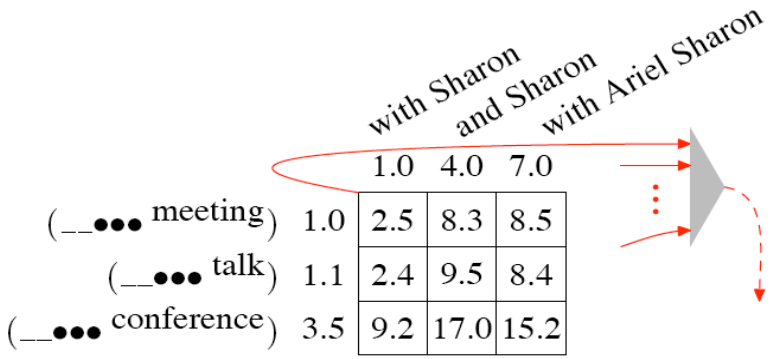
Pharaoh
(forward)



Cube Pruning
(backward)

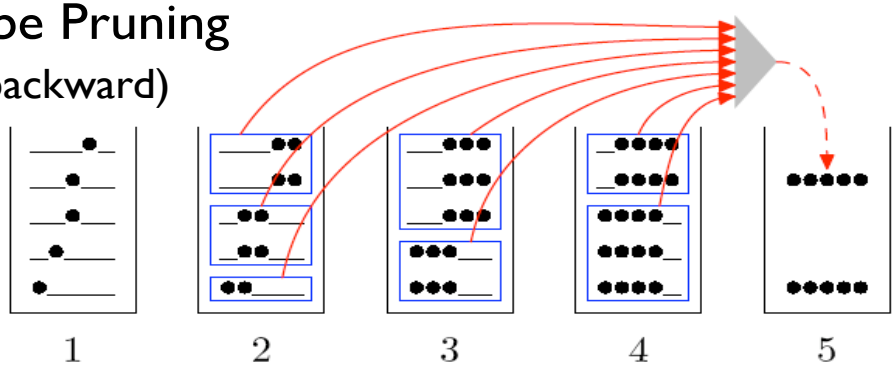


Cube Pruning

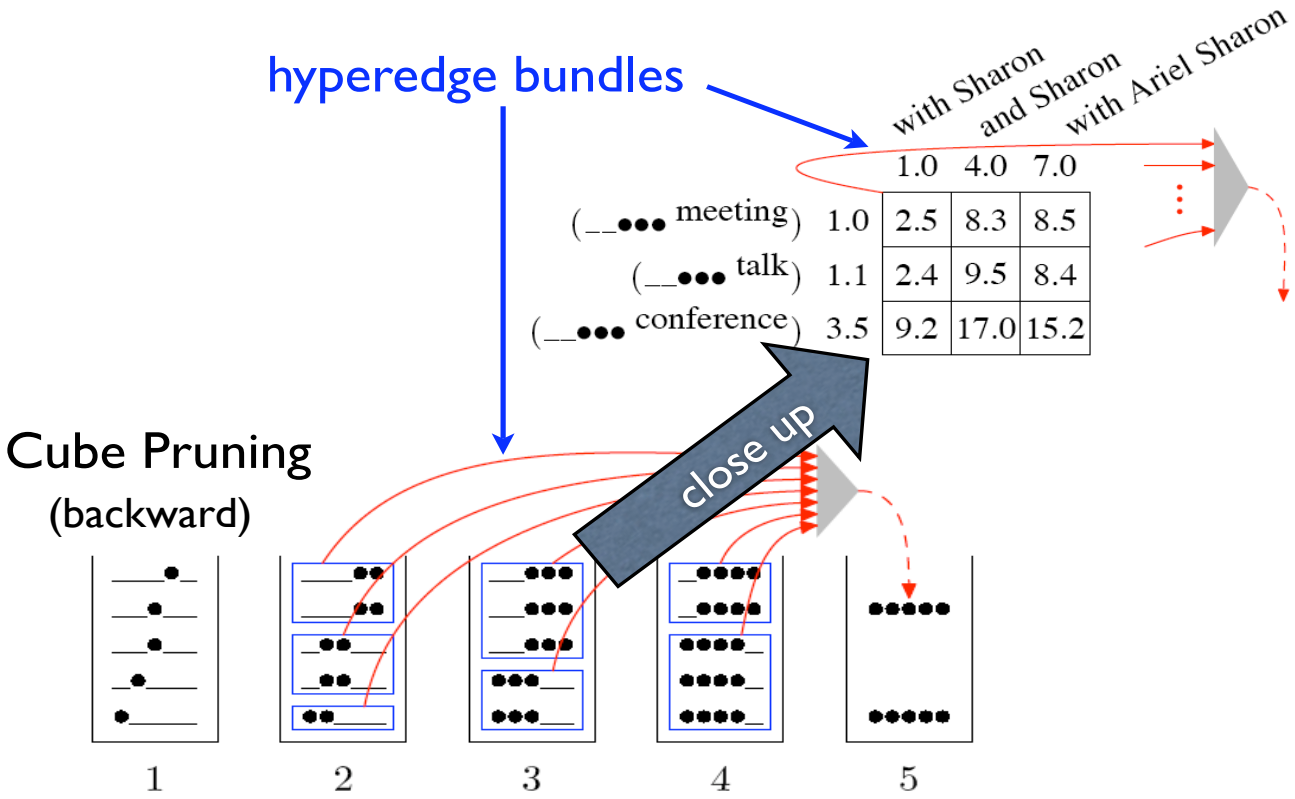


Cube Pruning

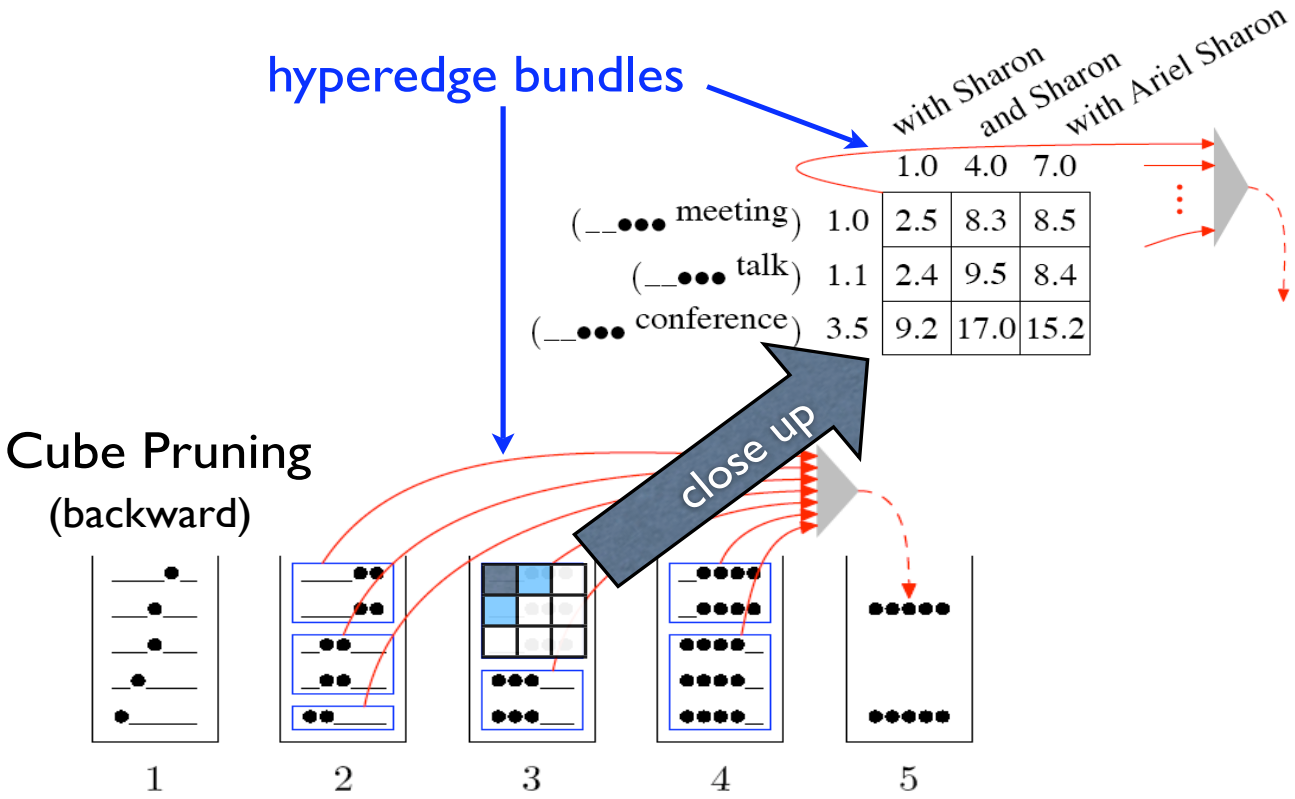
(backward)



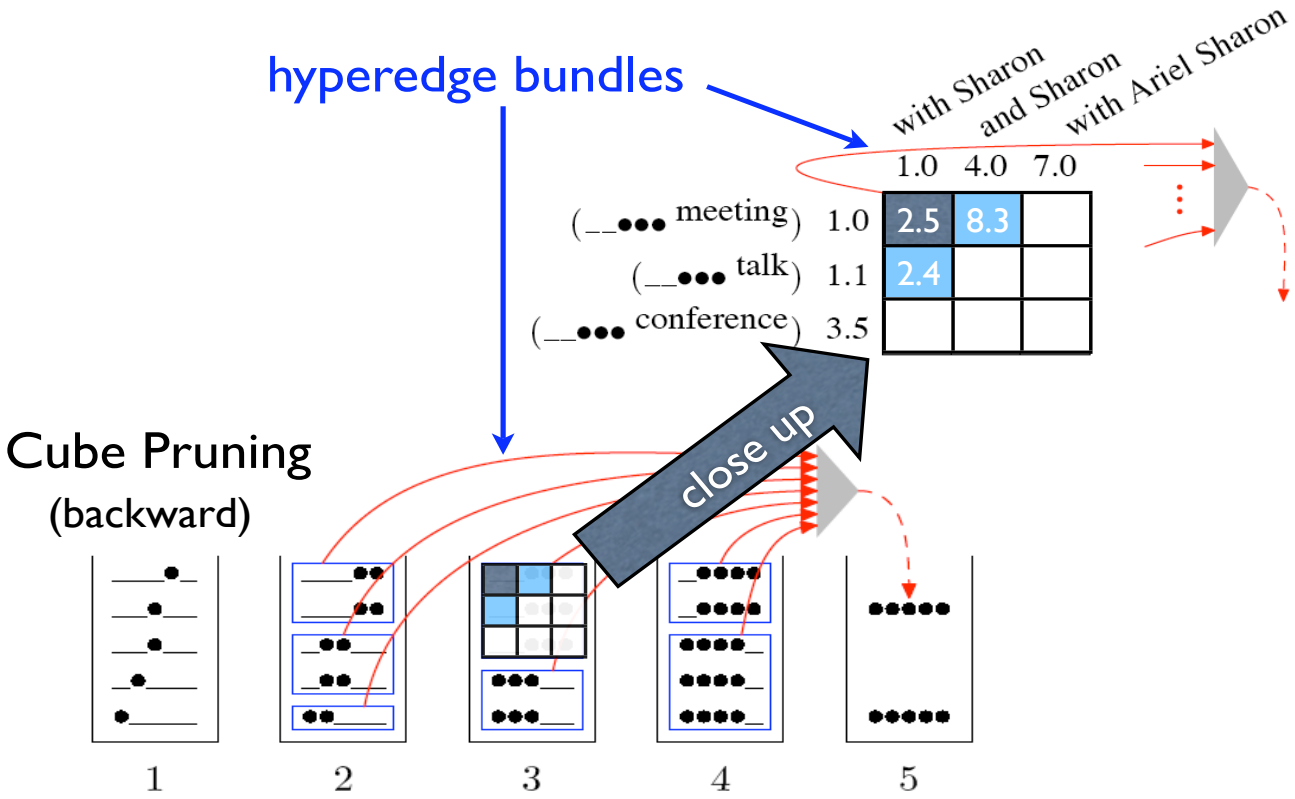
Cube Pruning



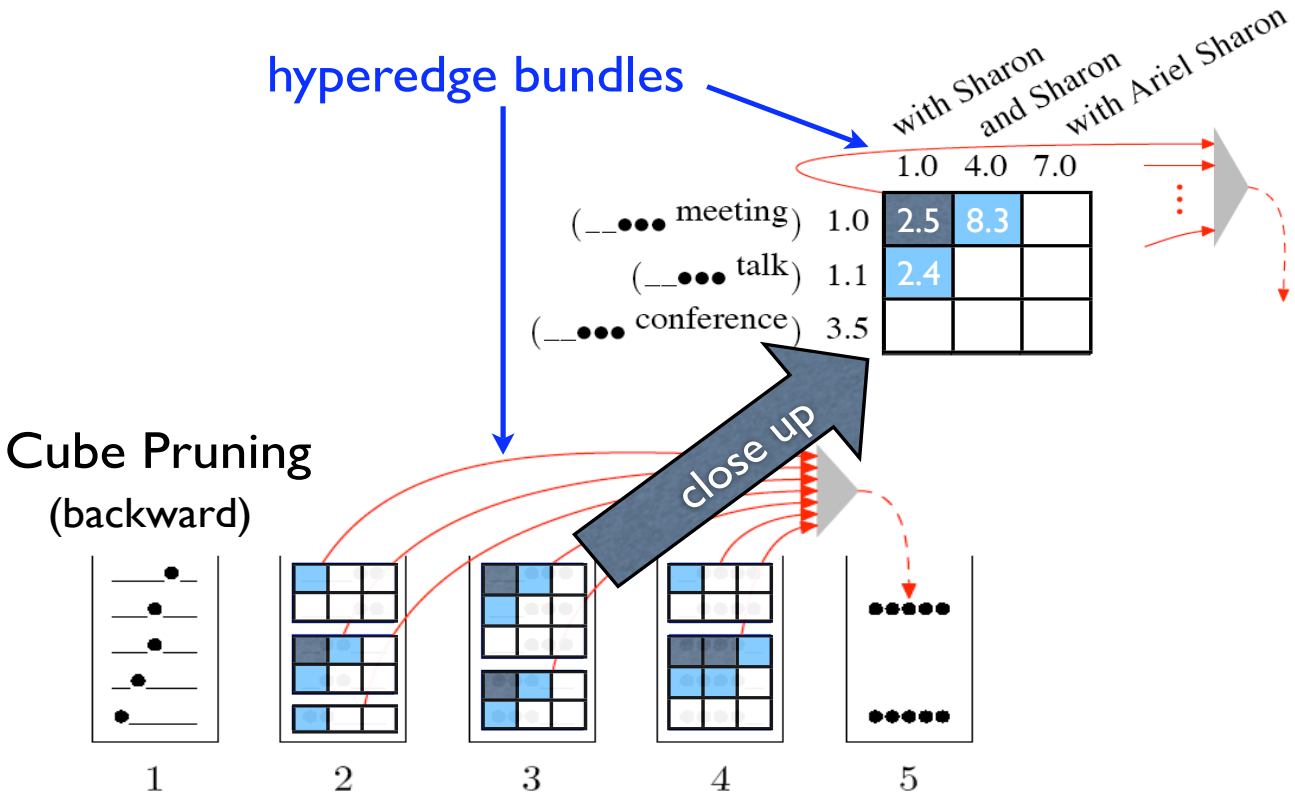
Cube Pruning



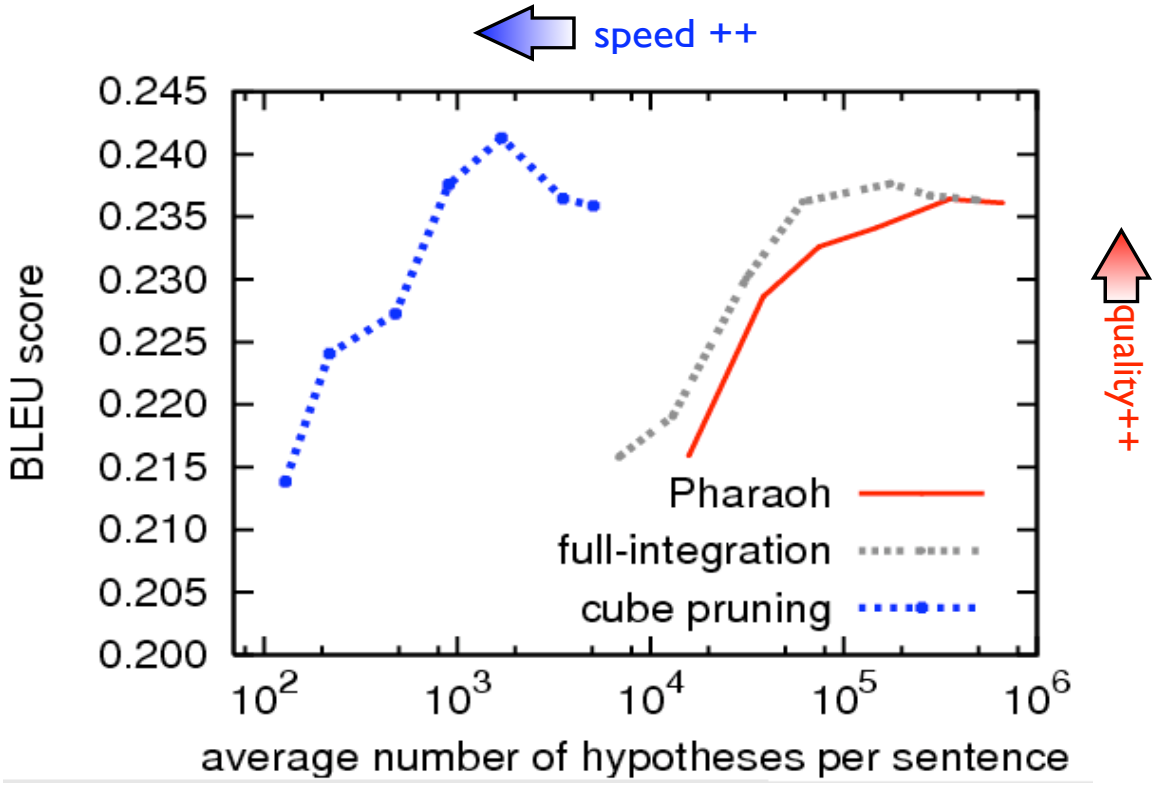
Cube Pruning



Cube Pruning



Cube Pruning



System Combination

- Can we improve translation quality by combining results from two (or more) decoders?
  - Promising! (see also earlier discussions of translation model and language model interpolation)
- Two approaches to system combination
  - Select the 'best' translation from any of the systems
  - Generate a new translation based on the translations from the individual systems
- Quality can depend on the degree to which the combination procedure has access to the search spaces of the individual systems
  - *(1.)* Translation string only $\rightarrow$ LM rescoring/voting
  - *(2.)* N-best translation strings only $\rightarrow$ MBR style combination
  - *(3.)* N-best translation derivations: $\rightarrow$ rescore based on weighted overlap between derivation steps
  - *(4.)* N-best translation derivations with feature weights: $\rightarrow$ rescore based on weighted overlap between features used

Recap

- Decoding as search
  - Search space representation
  - Partial hypotheses as states
  - Transition costs
  - Viterbi approximation
- Decoding complexity
  - DP decoding and recombination
- Pruning
  - Heuristic pruning
  - Beam search pruning and future cost estimation
  - Multi-stack decoding
- N-best lists and Minimum Bayes Risk
- Search errors vs. model errors
- Cube Pruning
- System combination