CS447: Natural Language Processing

http://courses.engr.illinois.edu/cs447

Lecture 23: Phrase-based MT (corrected)

Julia Hockenmaier

juliahmr@illinois.edu

3324 Siebel Center

Recap:
IBM models for MT

The IBM models

Use the noisy channel (Bayes rule) to get the best (most likely) target translation **e** for source sentence **f**:

$$\boxed{\arg\max_{\mathbf{e}} P(\mathbf{e}|\mathbf{f}) \quad = \quad \arg\max_{\mathbf{e}} P(\mathbf{f}|\mathbf{e})P(\mathbf{e})}$$

noisy channel

The translation model $P(\mathbf{f} \mid \mathbf{e})$ requires alignments **a**

$$\boxed{P(\mathbf{f}|\mathbf{e}) \quad = \quad \sum_{\mathbf{a}\in\mathcal{A}(\mathbf{e},\mathbf{f})} P(\mathbf{f},\mathbf{a}|\mathbf{e})}$$

marginalize (=sum) over all alignments a

Generate **f** and the alignment **a** with $P(\mathbf{f}, \mathbf{a} \mid \mathbf{e})$:

$$P(\mathbf{f},\mathbf{a}|\mathbf{e}) \quad = \quad \boxed{\underbrace{P(m|\mathbf{e})}_{\text{Length: } |\mathbf{f}|=m}} \prod_{j=1}^{m} \boxed{\underbrace{P(a_j|a_{1..j-1}, f_{1..j-1}, m, \mathbf{e})}_{\text{Word alignment } a_j}} \boxed{\underbrace{P(f_j|a_{1..j} f_{1..j-1}, \mathbf{e}, m)}_{\text{Translation } f_j}}$$

m = #words in fj

probability of alignment aj

probability of word fj

Representing word alignments

		1	2	3	4	5	6	7	8
		Marie	a	traversé	le	lac	à	la	nage
0	NULL						■	■	
1	Mary	■							
2	swam								■
3	across		■	■					
4	the				■				
5	lake					■			

Position	1	2	3	4	5	6	7	8
Foreign	Marie	a	traversé	le	lac	à	la	nage
Alignment	1	3	3	4	5	0	0	2

Every source word **f[i]** is aligned to one target word **e[j]** (incl. NULL).
We represent alignments as a vector **a** (of the same length as the source) with **a[i] = j**

IBM model 1: Generative process

For each target sentence $\mathbf{e} = e_1..e_n$ of length n:

0	1	2	3	4	5
NULL	**Mary**	**swam**	**across**	**the**	**lake**

1. **Choose a length** m for the source sentence (e.g $m = 8$)

Position	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**

2. **Choose an alignment** $\mathbf{a} = a_1 ... a_m$ for the source sentence

Each a_j corresponds to a word e_i in $\mathbf{e}$: $0 \leq a_j \leq n$

Position	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Alignment	**1**	**3**	**3**	**4**	**5**	**0**	**0**	**2**

3. **Translate each target word** e_{aj} into the source language

Position	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Alignment	**1**	**3**	**3**	**4**	**5**	**0**	**0**	**2**
Translation	**Marie**	**a**	**traversé**	**le**	**lac**	**à**	**la**	**nage**

Expectation-Maximization (EM)

1. Initialize a first model, M_0

2. Expectation (E) step:
   Go through training data to gather expected counts
   $\langle \text{count}(lac, lake) \rangle$
3. Maximization (M) step:
   Use expected counts to compute a new model M_{i+1}
   $P_{i+1}(lac \mid lake) = \langle \text{count}(lac, lake) \rangle / \langle \sum w \ \text{count}(w, lake) \rangle$
4. Check for convergence:
   Compute log-likelihood of training data with M_{i+1}
   If the difference between new and old log-likelihood smaller than a threshold, stop. Else go to 2.

The E-step

Compute the expected count $\langle c(f,e|\mathbf{f},\mathbf{e})\rangle$:

$$\langle c(f,e|\mathbf{f},\mathbf{e})\rangle = \sum_{\mathbf{a}\in\mathcal{A}(\mathbf{f},\mathbf{e})} \boxed{P(\mathbf{a}|\mathbf{f},\mathbf{e})} \cdot \underbrace{\boxed{c(f,e|\mathbf{a},\mathbf{e},\mathbf{f})}}_{\text{How often are } f,e \text{ aligned in } \mathbf{a}?}$$

$$\boxed{P(\mathbf{a}|\mathbf{f},\mathbf{e})} = \frac{\boxed{P(\mathbf{a},\mathbf{f}|\mathbf{e})}}{P(\mathbf{f}|\mathbf{e})} = \frac{\boxed{P(\mathbf{a},\mathbf{f}|\mathbf{e})}}{\sum_{\mathbf{a}'} \boxed{P(\mathbf{a}',\mathbf{f}|\mathbf{e})}}$$

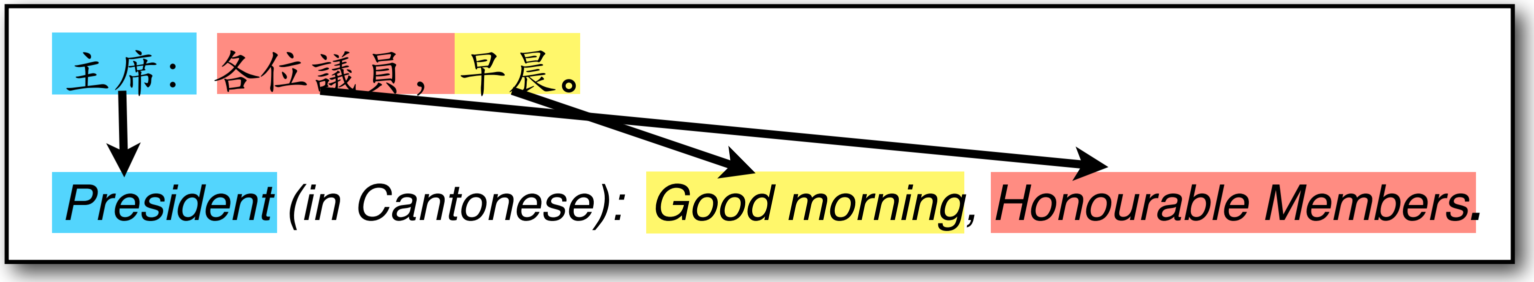
$$\boxed{P(\mathbf{a},\mathbf{f}|\mathbf{e})} = \prod_j \boxed{P(f_j|e_{a_j})}$$

$$\boxed{\langle c(f,e|\mathbf{f},\mathbf{e})\rangle} = \sum_{\mathbf{a}\in\mathcal{A}(\mathbf{f},\mathbf{e})} \frac{\prod_j \boxed{P(f_j|e_{a_j})}}{\sum_{a'}\prod_j \boxed{P(f_j|e_{a'_j})}} \cdot c(f,e|\mathbf{a},\mathbf{e},\mathbf{f})$$

Phrase-based translation models

Phrase-based translation models

Assumption: fundamental units of translation are **phrases**:



Phrase-based model of $P(F \mid E)$:

1. Split target sentence deterministically into phrases $ep_1 \ldots ep_n$

2. Translate each target phrase ep_i into source phrase fp_i with translation probability $\varphi(fp_i \mid ep_i)$

3. Reorder foreign phrases with distortion probability $d(a_i - b_{i-1}) = c^{|a_i - b_{i-1} - 1|}$

a_i = start position of source phrase generated by e_i

b_{i-1} = end position of source phrase generated by e_{i-1}

Phrase-based models of $P(f \mid e)$

Split target sentence $e = e_{1..n}$ into phrases $\mathbf{ep}_1..\mathbf{ep}_N$:

[The green witch] [is] [at home] [this week]

Translate each target phrase $\mathbf{ep}_i$ into source phrase $\mathbf{fp}_i$ with **translation probability** $P(fp_i \mid ep_i)$:

***[The green witch]* = *[die grüne Hexe]*, ...**

Arrange the set of source phrases $\{ fp_i \}$ to get $\mathbf{s}$ with **distortion probability** $P(fp \mid \{ fp_i \})$:

[Diese Woche] [ist] [die grüne Hexe] [zuhause]

$$P(\mathbf{f} \mid \mathbf{e} = \langle ep_1, ..., ep_l \rangle) \quad = \quad \prod_i P(fp_i \mid ep_i) P(\mathbf{fp} \mid \{fp_i\})$$

Translation probability $P(fp_i \mid ep_i)$

Phrase translation probabilities can be obtained from a **phrase table:**

EP	**FP**	**count**
green witch	grüne Hexe	…
at home	zuhause	10534
at home	daheim	9890
is	ist	598012
this week	diese Woche	….

This requires **phrase alignment**

Word alignment

	Diese	Woche	ist	die	grüne	Hexe	zuhause
The				■			
green					■		
witch						■	
is			■				
at							
home							■
this	■						
week		■					

Phrase alignment

	Diese	Woche	ist	die	grüne	Hexe	zuhause
The				■	▨	▨	
green				▨	■	▨	
witch				▨	▨	■	
is			■				
at							▨
home							■
this	■	▨					
week	▨	■					

Obtaining phrase alignments

We'll skip over details, but here's the basic idea:

For a given parallel corpus (F-E)

1. Train **two word aligners**, (F→E and E→F)
2. Take the **intersection** of these alignments to get a **high-precision** word alignment
3. **Grow** these high-precision alignments until all words in both sentences are included in the alignment.

   Consider any pair of words in the **union** of the alignments, and incrementally add them to the existing alignments
4. Consider all phrases that are **consistent** with this improved word alignment

Decoding
(for phrase-based MT)

Phrase-based models of $P(f \mid e)$

Split target sentence $\boldsymbol{e}=e_{1..n}$ into phrases $\mathbf{ep}_1..\mathbf{ep}_N$:

[The green witch] [is] [at home] [this week]

Translate each target phrase $\mathbf{ep}_\mathrm{i}$ into source phrase $\mathbf{fp}_\mathrm{i}$ with **translation probability** $P(\boldsymbol{fp}_i \mid \boldsymbol{ep}_i)$:

[The green witch] = [die grüne Hexe], ...

Arrange the set of source phrases $\{\boldsymbol{fp}_i\}$ to get $\mathbf{s}$ with **distortion probability** $P(\boldsymbol{fp} \mid \{\boldsymbol{fp}_i\})$:

[Diese Woche] [ist] [die grüne Hexe] [zuhause]

$$P(\mathbf{f} \mid \mathbf{e} = \langle ep_1, ..., ep_l \rangle) \quad = \quad \prod_i P(fp_i \mid ep_i) P(\mathbf{fp} \mid \{fp_i\})$$

Translating

How do we translate a foreign sentence (e.g. *“Diese Woche ist die grüne Hexe zuhause”*) into English?

- We need to find $\hat{e} = argmax_e P(f \mid e)P(e)$
- There is an exponential number of candidate translations e
- But we can look up phrase translations ep and $P(fp \mid ep)$ in the phrase table:

diese	**Woche**	**ist**	**die**	**grüne**	**Hexe**	**zuhause**
this 0.2	week 0.7	is 0.8	the 0.3	green 0.3	witch 0.5	home 1.00
these 0.5			the green 0.4		sorceress 0.6	
this week 0.6				green witch 0.7		
is this week 0.4			the green witch 0.7			

Generating a (random) translation

1. Pick the first Target phrase $\boldsymbol{ep}_1$ from the candidate list.

$$P := P_{LM}(<s>\ \boldsymbol{ep}_1)P_{Trans}(\boldsymbol{fp}_1 \mid \boldsymbol{ep}_1)$$

E = *the*, *F*= *<….die…>*

2. Pick the next target phrase $\boldsymbol{ep}_2$ from the candidate list

$$P := P \times P_{LM}(\boldsymbol{ep}_2 \mid \boldsymbol{ep}_1)P_{Trans}(\boldsymbol{fp}_2 \mid \boldsymbol{ep}_2)$$

E = *the green witch*, *F* = *<….die grüne Hexe...>*

3. Keep going: pick target phrases $\boldsymbol{ep}_\mathrm{i}$ until the entire source sentence is translated

$$P := P \times P_{LM}(\boldsymbol{ep}_\mathrm{i} \mid \boldsymbol{ep}_{1\ldots\mathrm{i}-1})P_{Trans}(\boldsymbol{fp}_\mathrm{i} \mid \boldsymbol{ep}_\mathrm{i})$$

E = *the green witch is*, *F* = *<….ist die grüne Hexe...>*

diese	Woche	ist	die	grüne	Hexe	zuhause
this 0.2	week 0.7	③ is 0.8	① the 0.3	green 0.3	witch 0.5	⑤ at home 0.5
these 0.5			the green 0.4		sorceress 0.6	
④ this week 0.6				② green witch 0.7		
is this week 0.4			the green witch 0.7			

Finding the best translation

How can we find the *best* translation efficiently?

- There is an exponential number of possible translations.

We will use a *heuristic* search algorithm

- We cannot guarantee to find the best (= highest-scoring) translation, but we're likely to get close.

We will use a *"stack-based"* decoder

- (If you've taken Intro to AI: this is A* ("A-star") search)
- We will score partial translations based on how good we expect the corresponding completed translation to be.
  - Or, rather: we will score partial translations on how **bad** we expect the corresponding complete translation to be.
  - That is, our scores will be **costs (high=bad, low=good)**

Scoring partial translations

Assign expected costs to *partial* translations (E, F):

$$expected\_cost(E,F) = current\_cost(E,F) + future\_cost(E,F)$$

The current cost is based on the score of the partial translation (E, F)

e.g. $current\_cost(E,F) = \log P(E)P(F \mid E)$

The (estimated) future cost is a **lower bound** on the actual cost of completing the partial translation (E, F):

$$true\_cost(E,F) \quad (= current\_cost(E,F) + actual\_future\_cost(E,F))$$
$$\geq expected\_cost(E,F) \quad (= current\_cost(E,F) + est\_future\_cost(E,F))$$

because $actual\_future\_cost(E,F) \geq est\_future\_cost(E,F)$

(The estimated future cost ignores the distortion cost)

Stack-based decoding

Maintain a **priority queue** (=’stack’) of **partial translations** (hypotheses) with their **expected costs**.

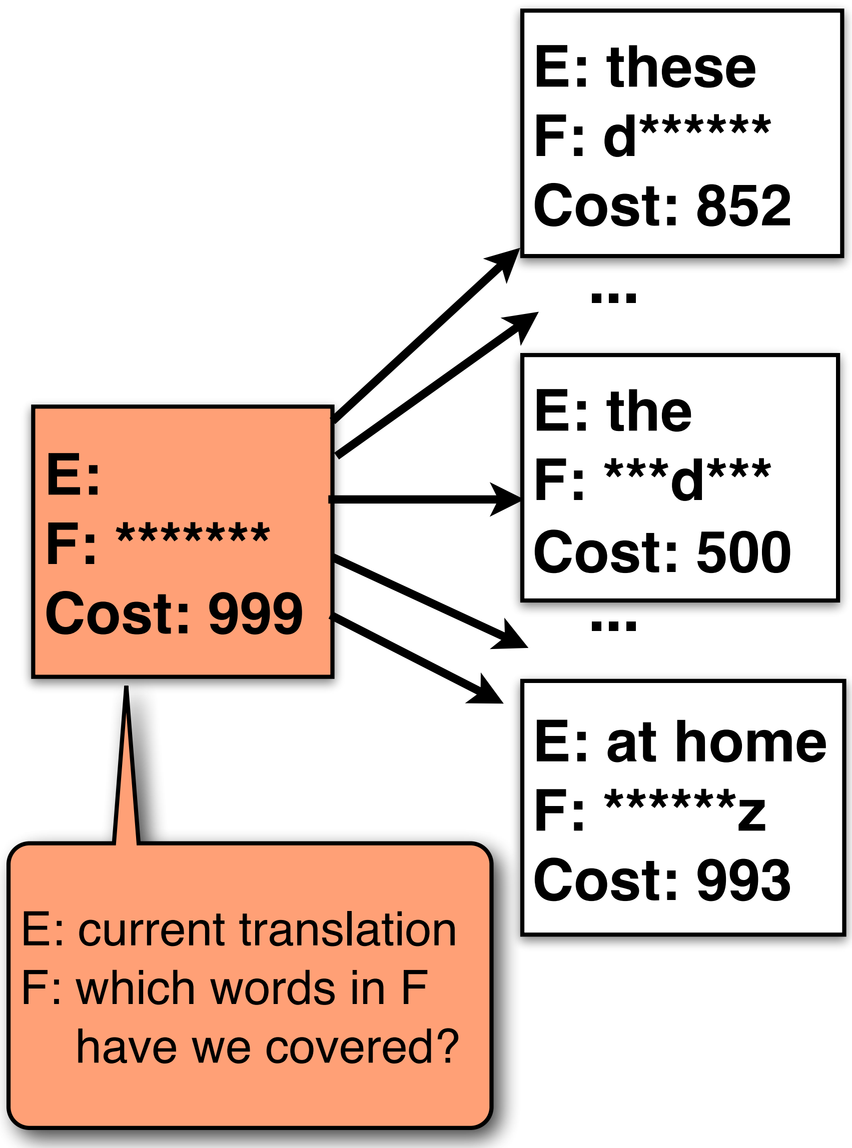
Each element on the stack is **open** (we haven’t yet pursued this hypothesis) or **closed** (we have already pursued this hypothesis)

At each step:

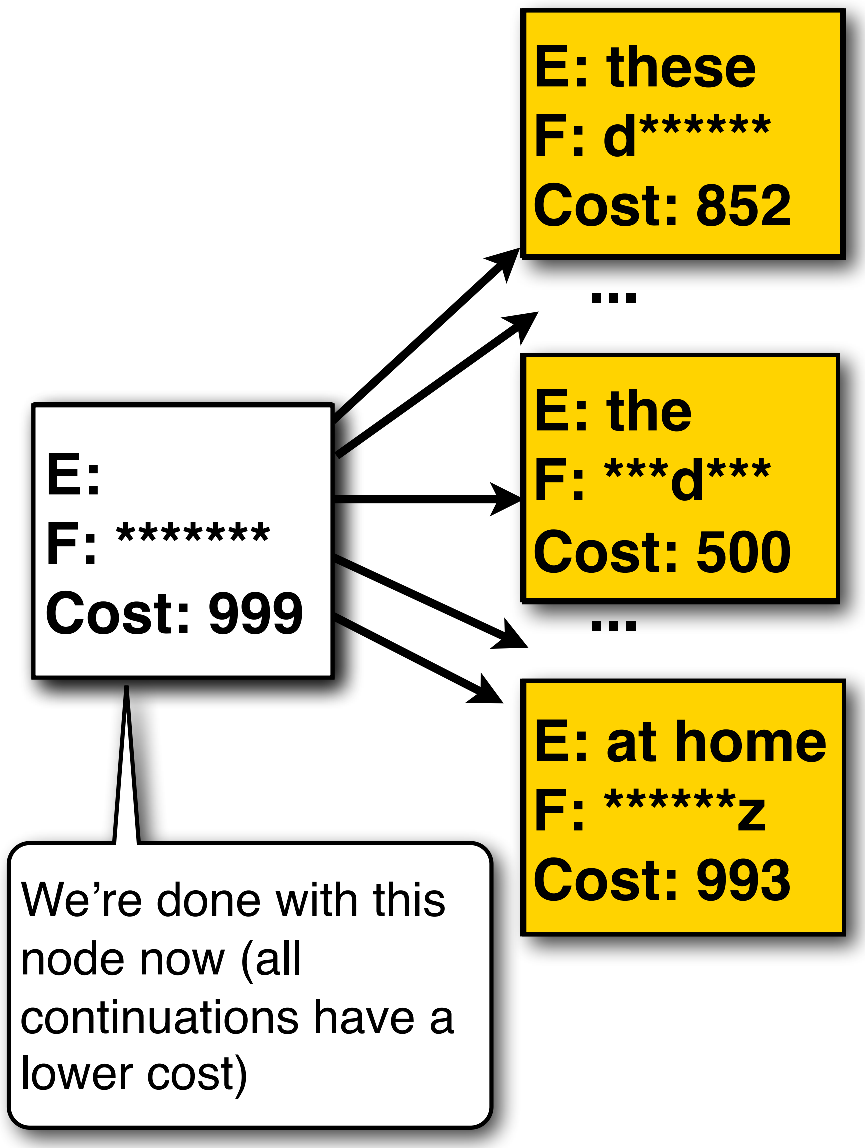
- **Expand** the best open hypothesis (the open translation with the lowest expected cost) in all possible ways.
- These new translations become new **open elements** on the stack.
- **Close** the best open hypothesis.

Additional Pruning (*n*-best / beam search):
Only keep the *n* best open hypotheses around

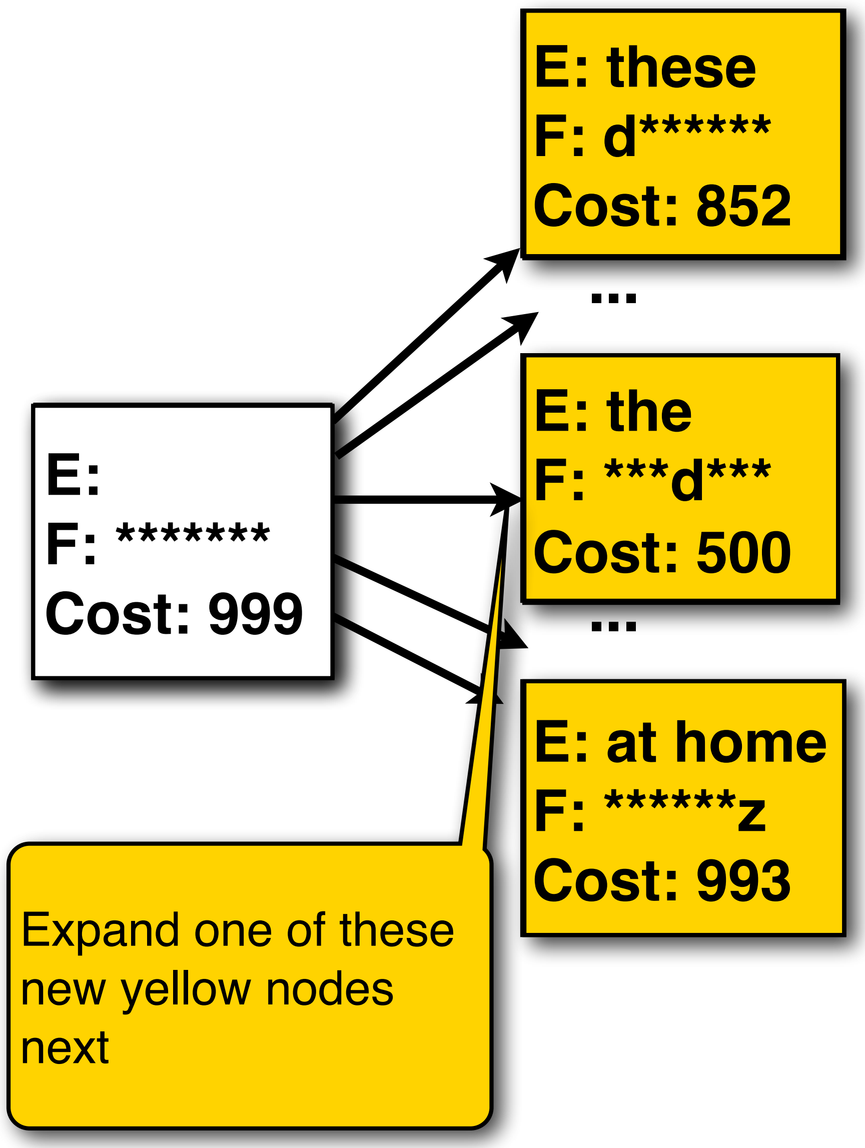
Stack-based decoding



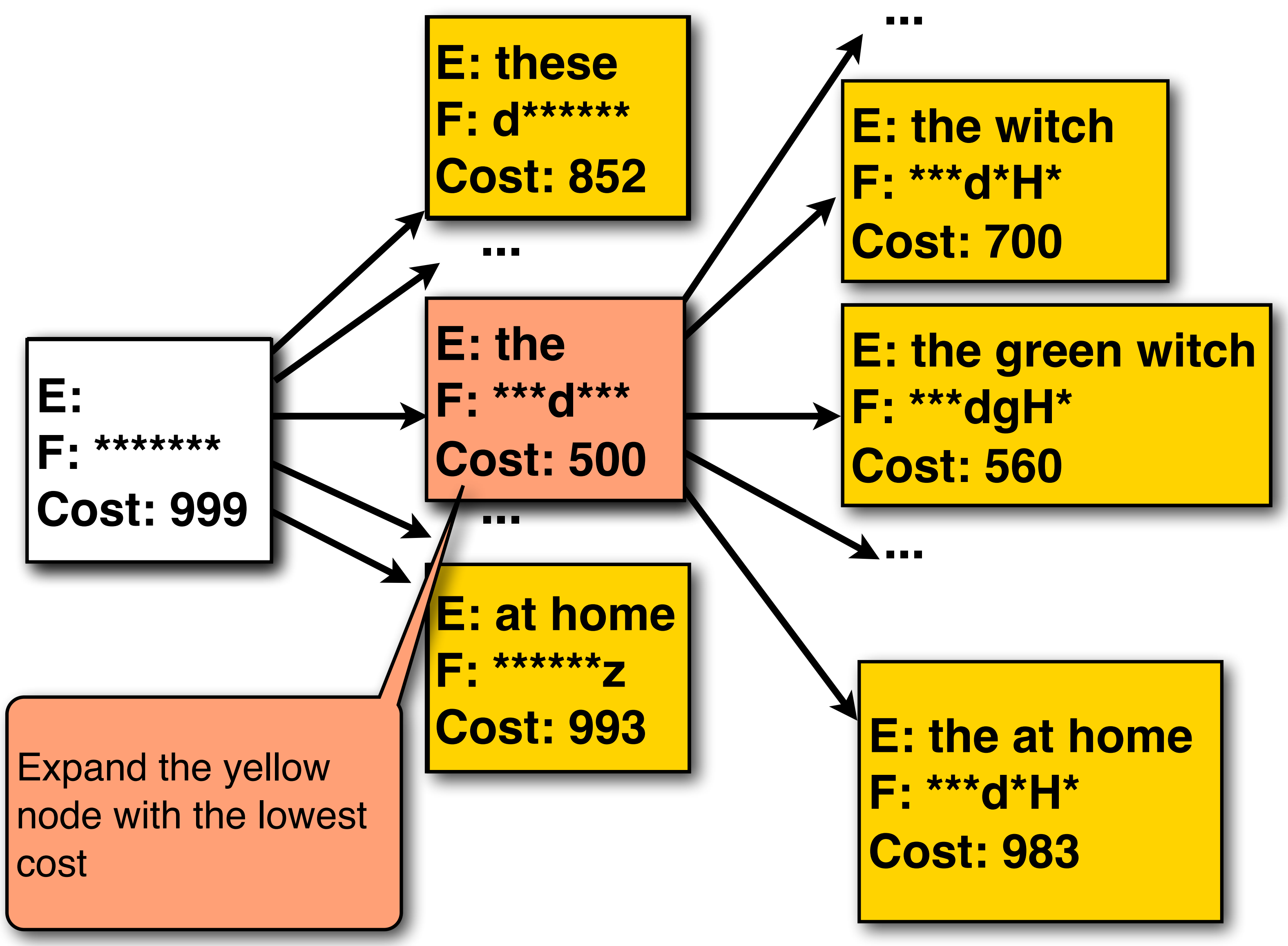
Stack-based decoding



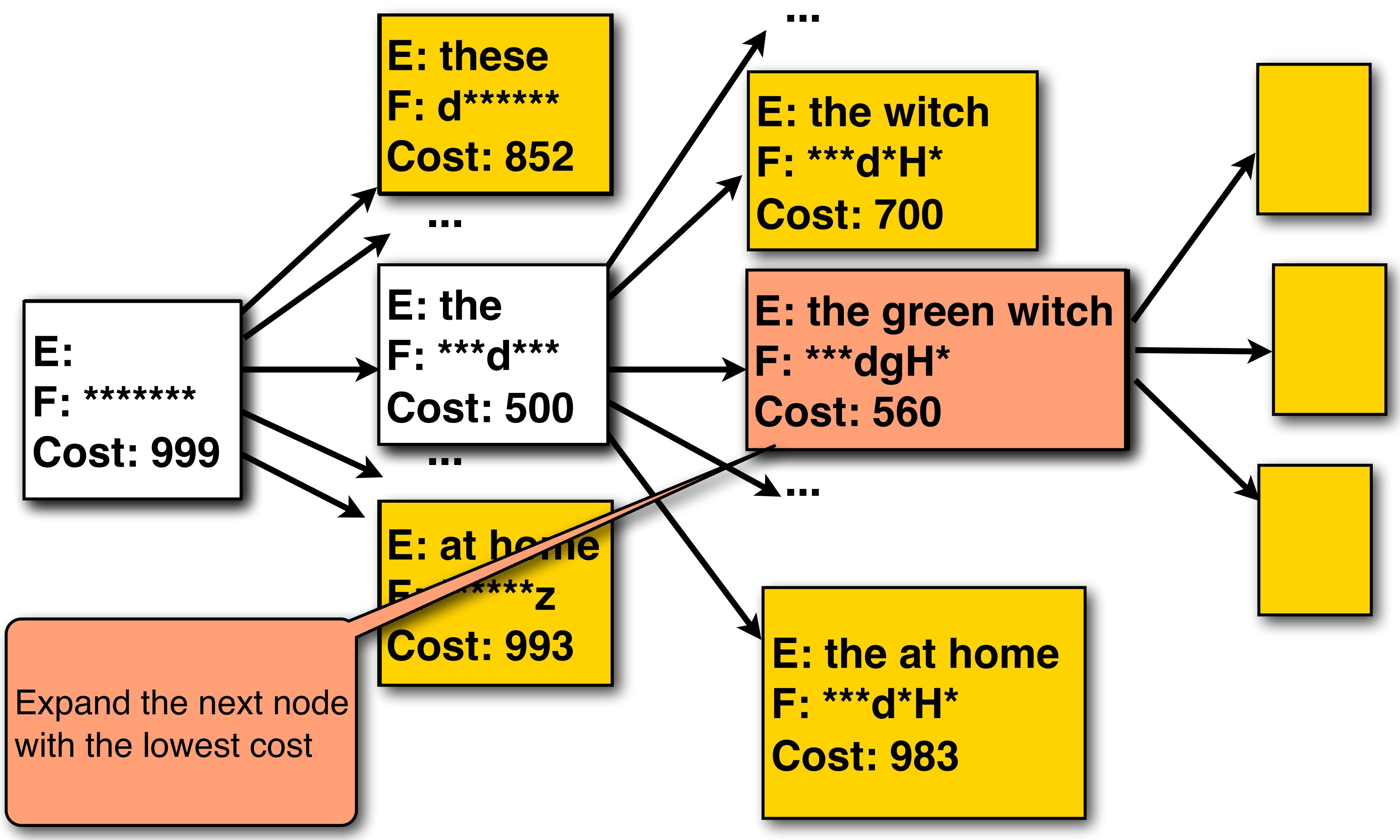
Stack-based decoding



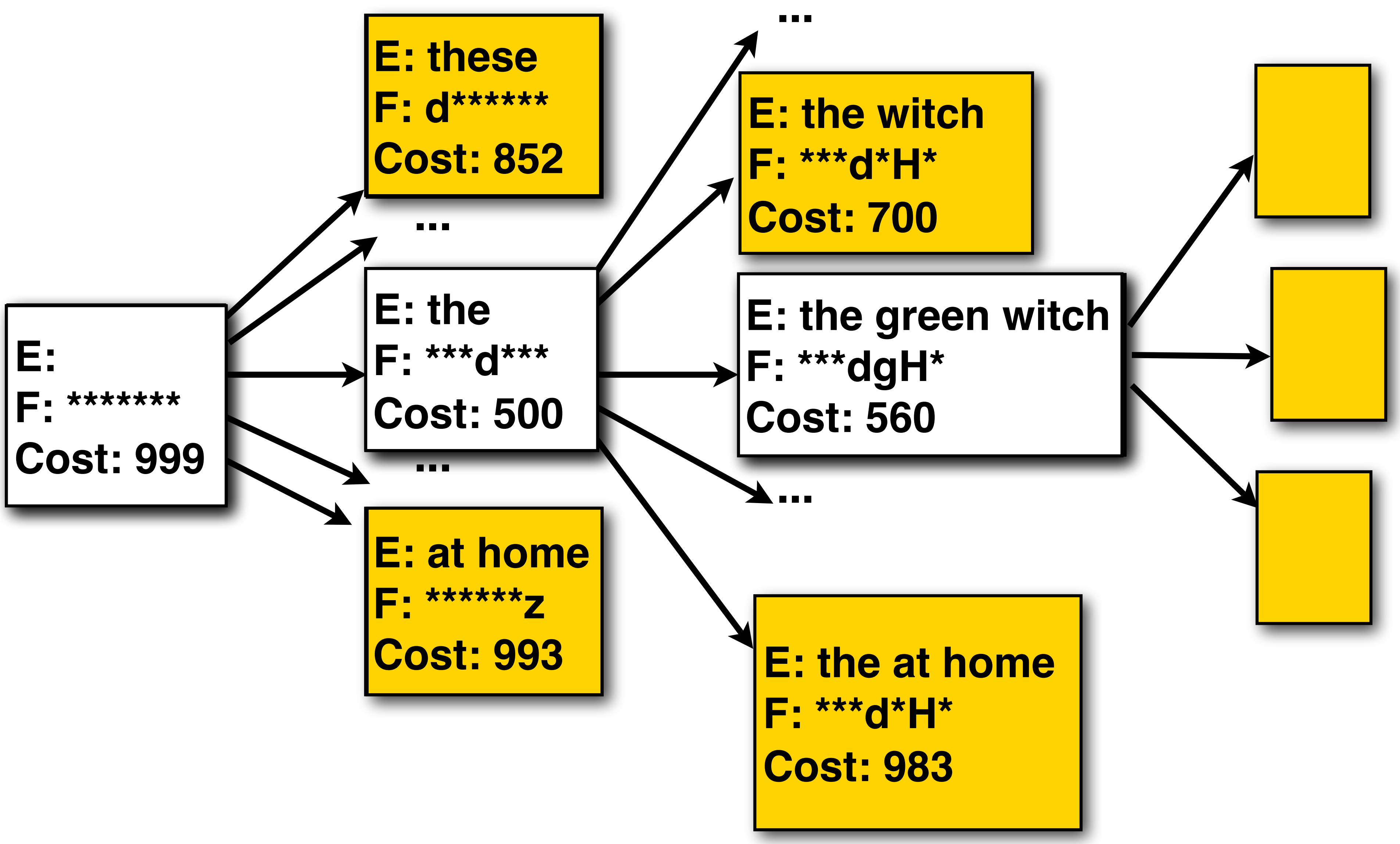
Stack-based decoding



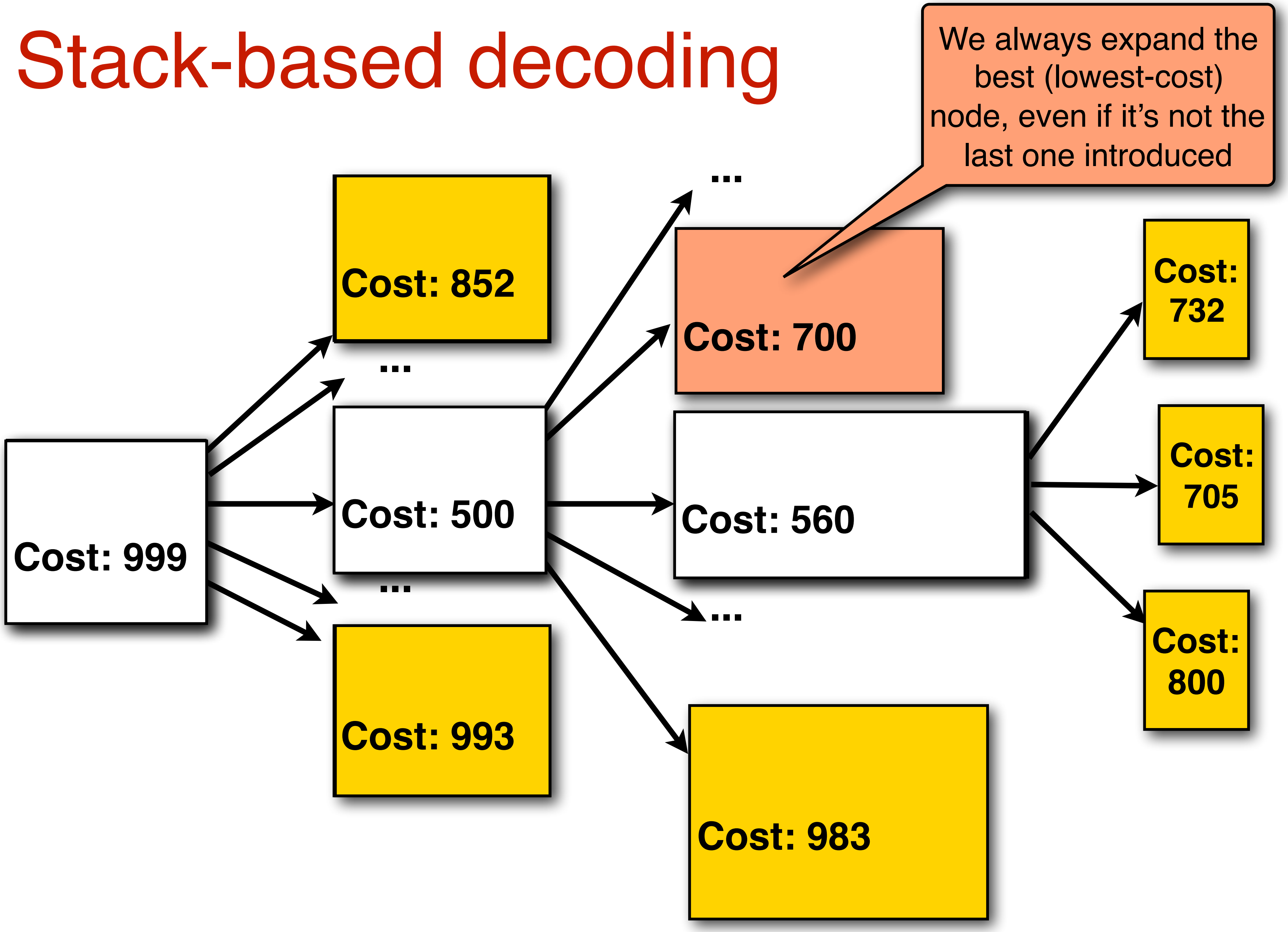
Stack-based decoding



Stack-based decoding



Stack-based decoding



MT evaluation

Automatic evaluation: BLEU

Evaluate candidate translations against several reference translations.

C1: It is a guide to action which ensures that the military always obeys the commands of the party.
C2: It is to insure the troops forever hearing the activity guidebook that party direct

R1: It is a guide to action that ensures that the military will forever heed Party commands.
R2: It is the guiding principle which guarantees the military forces always being under the command of the Party.
R3: It is the practical guide for the army always to heed the directions of the party.

The **BLEU score** is based on **N-gram precision**:
How many n-grams in the candidate translation occur also in one of the reference translation?

BLEU details

For $n \in \{1,\ldots,4\}$, compute the (modified) **precision of all *n*-grams**:

$$Prec_n = \frac{\sum_{c \in C} \sum_{n\text{-gram} \in c} \mathrm{MaxFreq}_{\mathrm{ref}}(n\text{-gram})}{\sum_{c \in C} \sum_{\text{-gram} \in c} \mathrm{Freq}_{\mathrm{c}}(n\text{-gram})}$$

$\mathrm{MaxFreq}_{\mathrm{ref}}$(‘*the party*’) = max. count of ‘*the party*’ in **one** reference translation.

$\mathrm{Freq}_{\mathrm{c}}$(‘*the party*’) = count of ‘*the party*’ in candidate translation c.

Penalize short candidate translations by a **brevity penalty** BP

c = length (number of words) of the whole candidate translation corpus

r = Pick for each candidate the reference translation that is closest in length; sum up these lengths.

Brevity penalty $\mathrm{BP} = \exp(1 - c/r)$ for $c \leq r$; $\mathrm{BP} = 1$ for $c > r$

(BP ranges from e for $c=0$ to 1 for $c=r$)

BLEU score

The BLEU score is the geometric mean of the precision of the unigrams, bigrams, trigrams, quadrigrams,
weighted by the brevity penalty BP.

$$\mathbf{BLEU} = BP \times \exp\left(\frac{1}{N}\sum_{n=1}^{N}\log Prec_n\right)$$

Human evaluation

We want to know whether the translation is **"good" English**, and whether it is an **accurate translation** of the original.

- Ask human raters to judge the **fluency** and the **adequacy** of the translation (e.g. on a scale of 1 to 5)
- Correlated with fluency is accuracy on **cloze task**:
  Give rater the sentence with one word replaced by blank.
  Ask rater to guess the missing word in the blank.
- Similar to adequacy is **informativeness**
  Can you use the translation to perform some task
  (e.g. answer multiple-choice questions about the text)

Summary:
Machine Translation

Machine translation models

Current MT models all rely on statistics.

Many current models do estimate $P(\mathrm{E} \mid \mathrm{F})$ directly, but may use features based on language models (capturing $P(\mathrm{E})$) and IBM-style translation models $(\mathrm{P}(\mathrm{F} \mid \mathrm{E}))$ internally.

There are a number of syntax-based models, e.g. using synchronous context-free grammars, which consist of pairs of rules for the two languages in which each RHS NT in language A corresponds to a RHS NT in language B:

Language A: XP → YP ZP   Language B: XP → ZP YP

More recent developments

Neural network-based approaches:

Recurrent neural networks (RNN) can model sequences (e.g. strings, sentences, etc.)

Use one RNN (the encoder) to process the input in the source language

Pass its output to another RNN (the decoder) to generate the output in the target language

See e.g. http://www.tensorflow.org/tutorials/seq2seq/index.md#sequence-to-sequence_basics