Natural Language Processing: MT conclusion & Language models

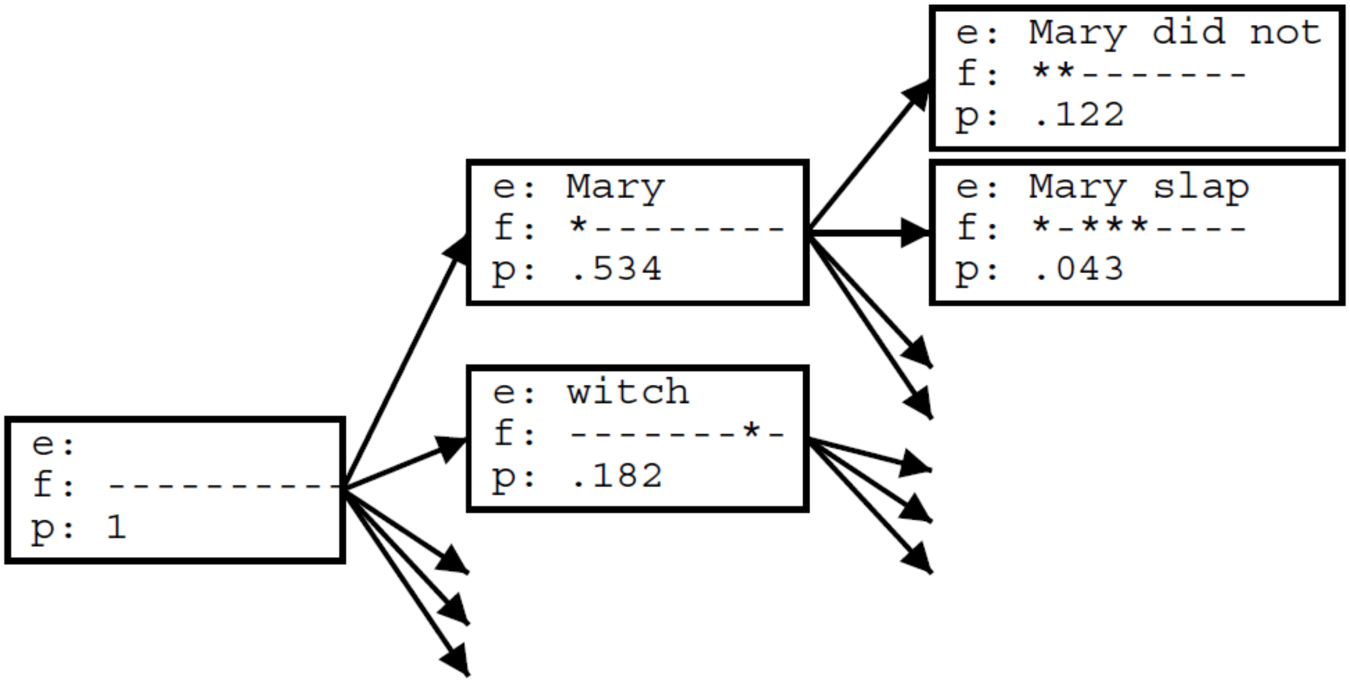
Christopher Manning

Borrows some slides from Kevin Knight and Dan Klein

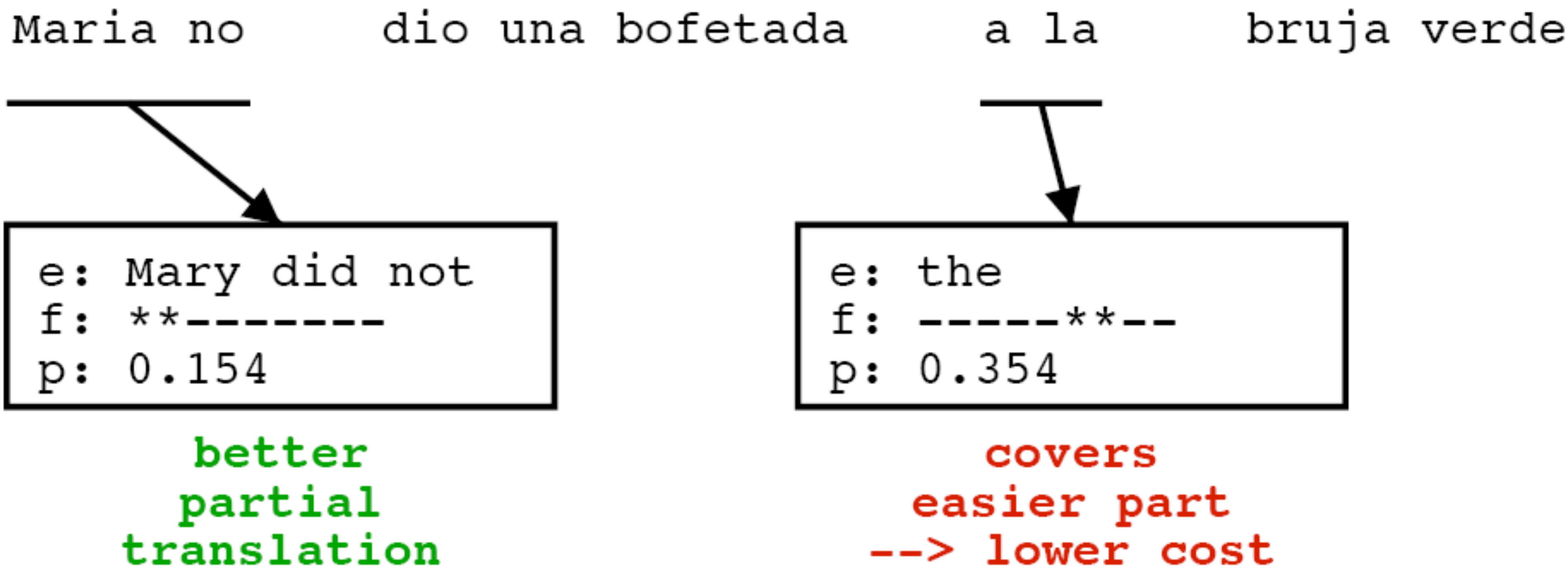
Lecture Plan

1. **Last bits of Machine Translation** [15 mins]
2. Language Models [15 mins]
3. Tuning and Evaluating Models [5 mins]
4. Final Project Discussion

Recall that we were searching for a good translation …



Pruning: Beams + Forward Costs



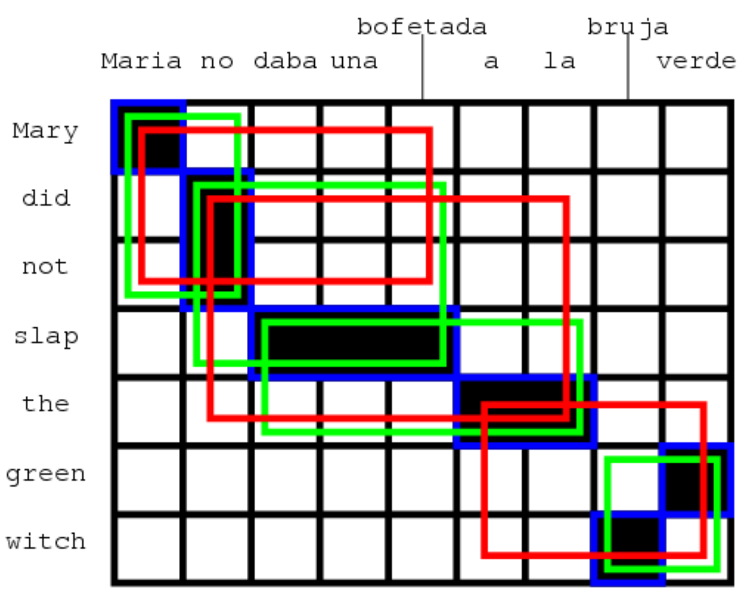
- Problem: easy partial analyses are cheaper
  - Solution 1: use beams per foreign subset
  - Solution 2: estimate forward costs (A*-like)

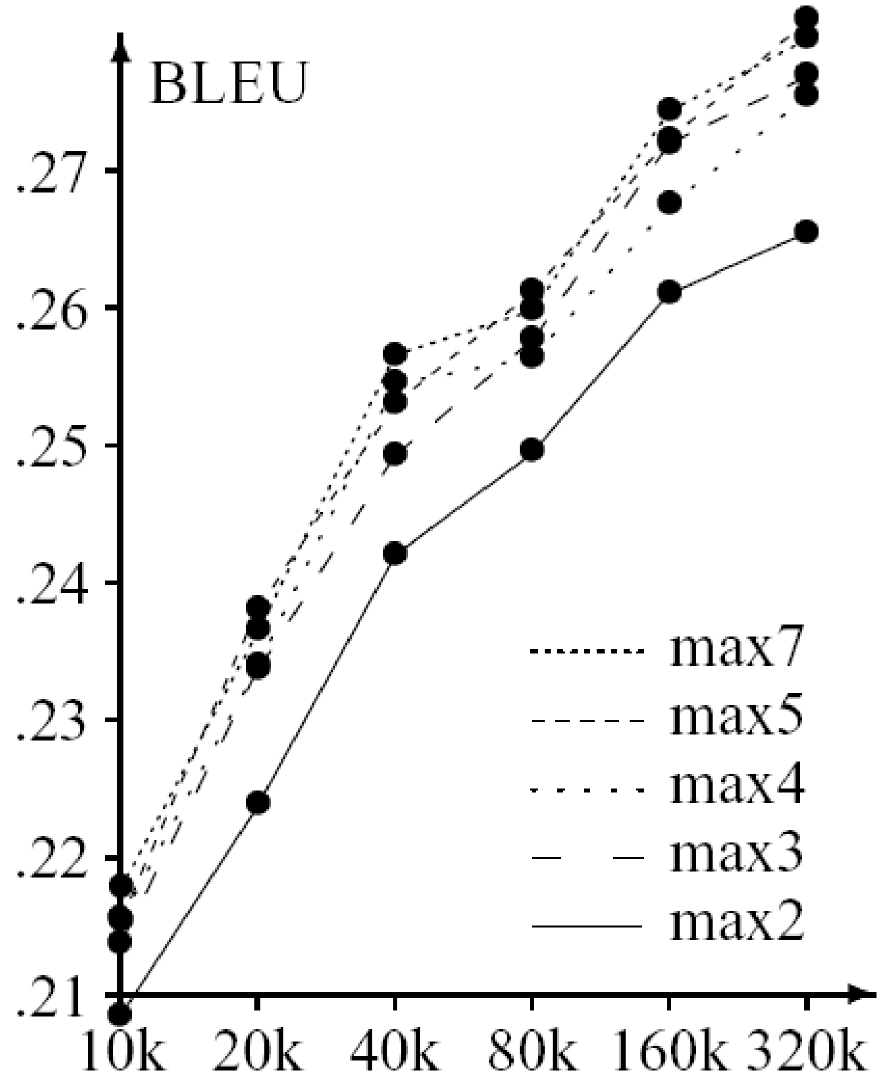
“Distortion”

- If our model were great, we’d let it rearrange phrases as much as it wants to
- In practice, that make translations **slow** and **bad**
- Commonly people put a hard limit on the size of reorderings
  - We do this in Phrasal in PA1

Phrase Size

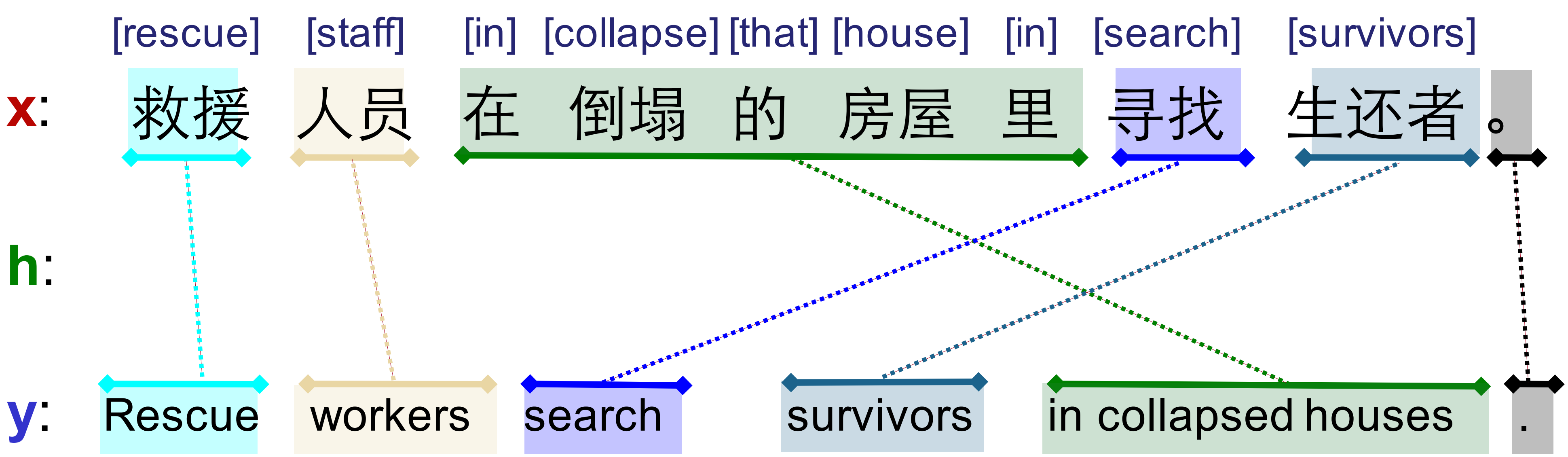
- Phrases help
  - But long ones often don't help much
  - Why should this be?





Local syntax in phrase-based systems

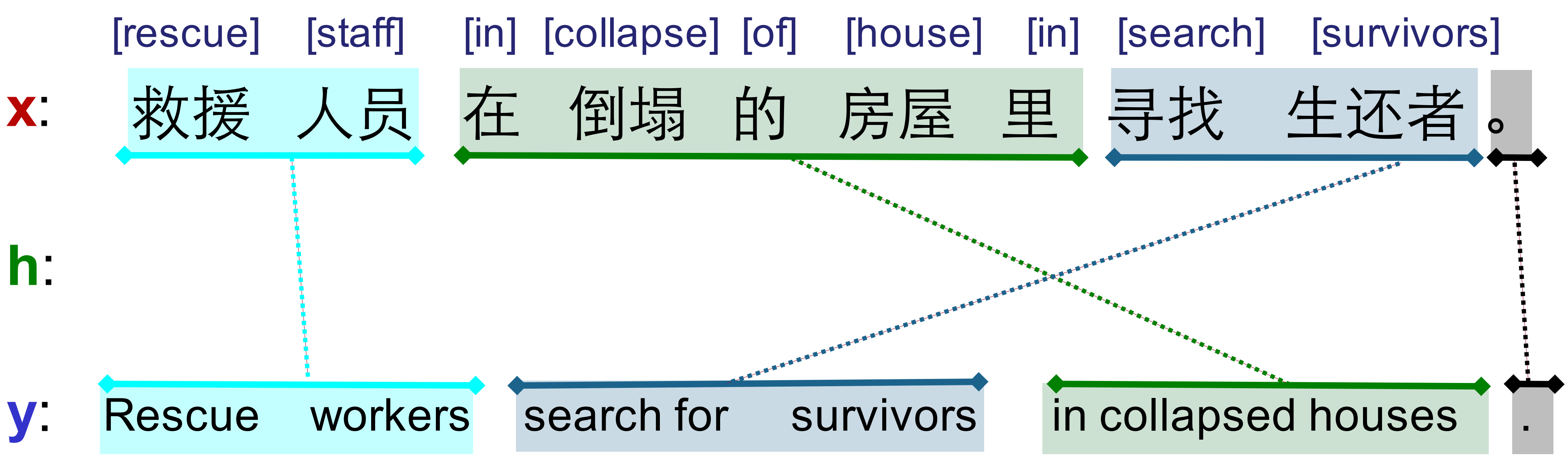
[Och et al., 1999; Och and Ney; 2004]



Phrases capture multi-word expressions,
help select correct function words,
and enable local reorderings.

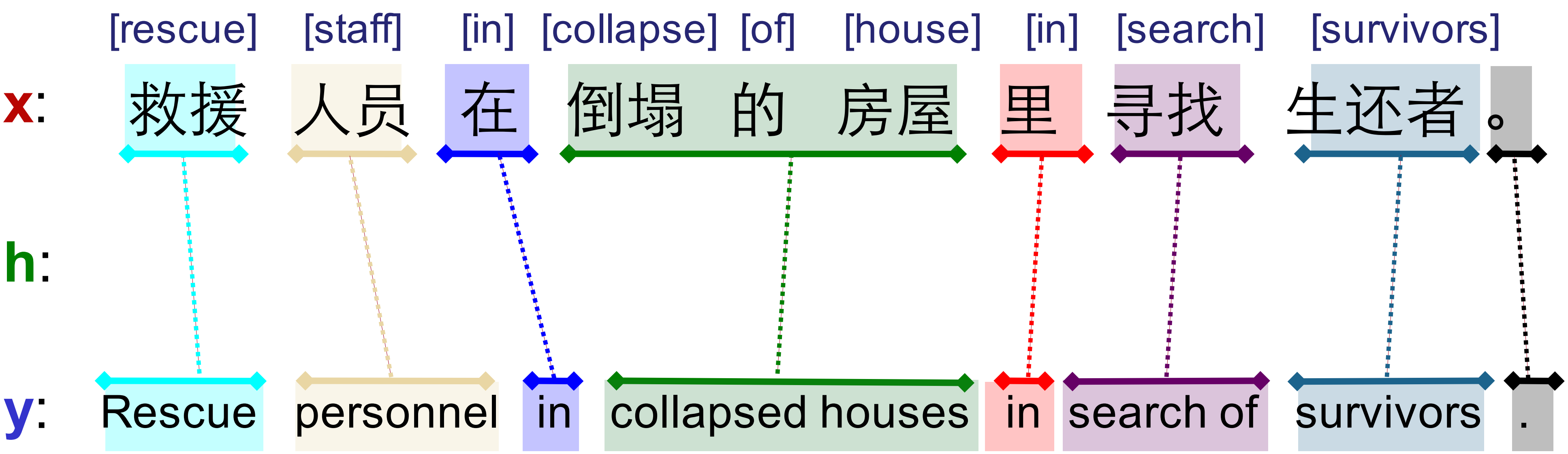
Local syntax in phrase-based systems

[Och et al., 1999; Och and Ney; 2004]



Phrases capture multi-word expressions,
help select correct function words (e.g., now also “for”),
and enable local reorderings.

Phrase-based models at test time



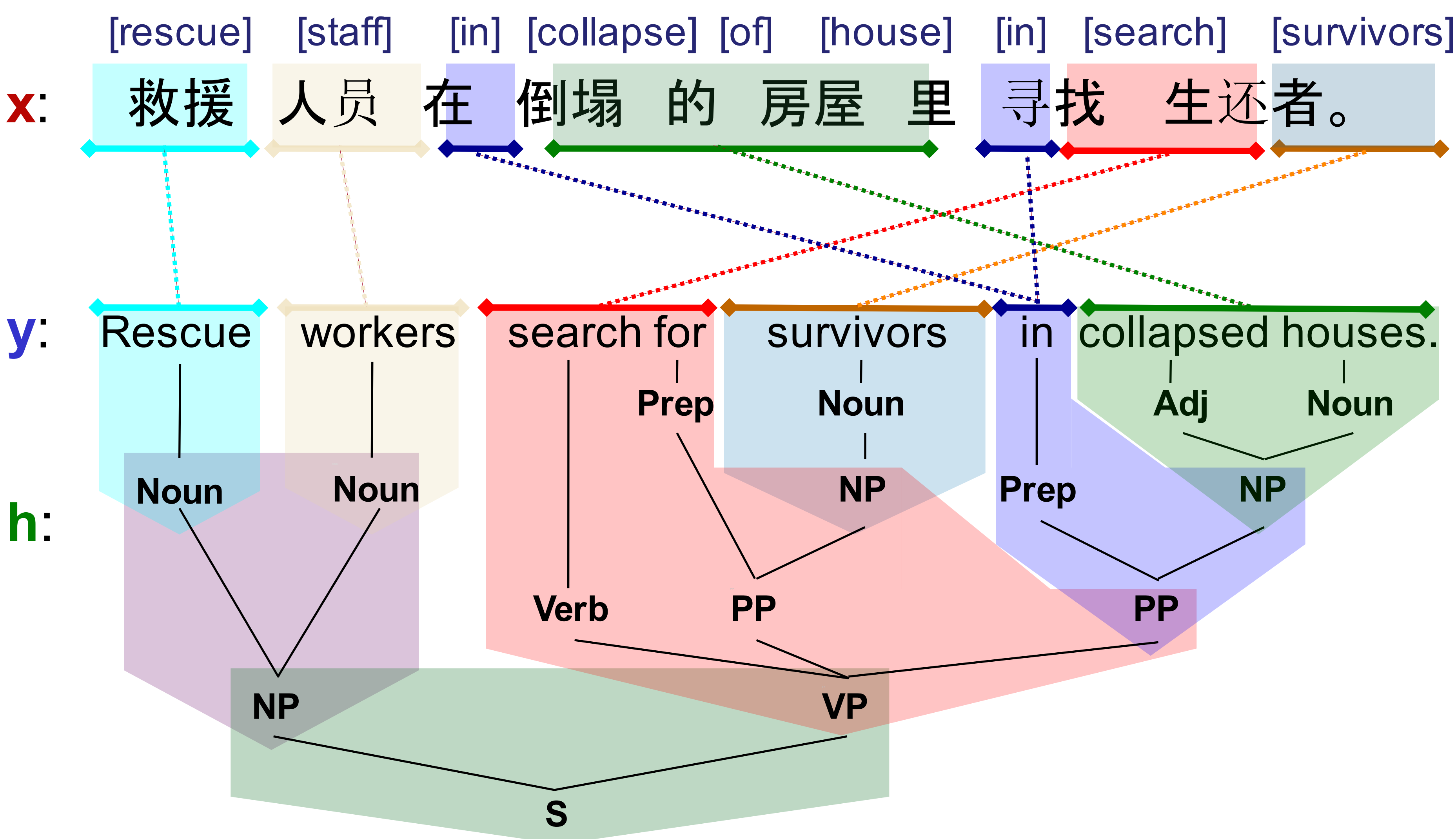
Google translate 's actual output, 2010

Oct 2015 output: Rescue workers in collapsed buildings in search of survivors.

Long test phrases are often unseen in training.
Short phrases yield poor translations.
Need a more effective model to account for non-local dependencies!

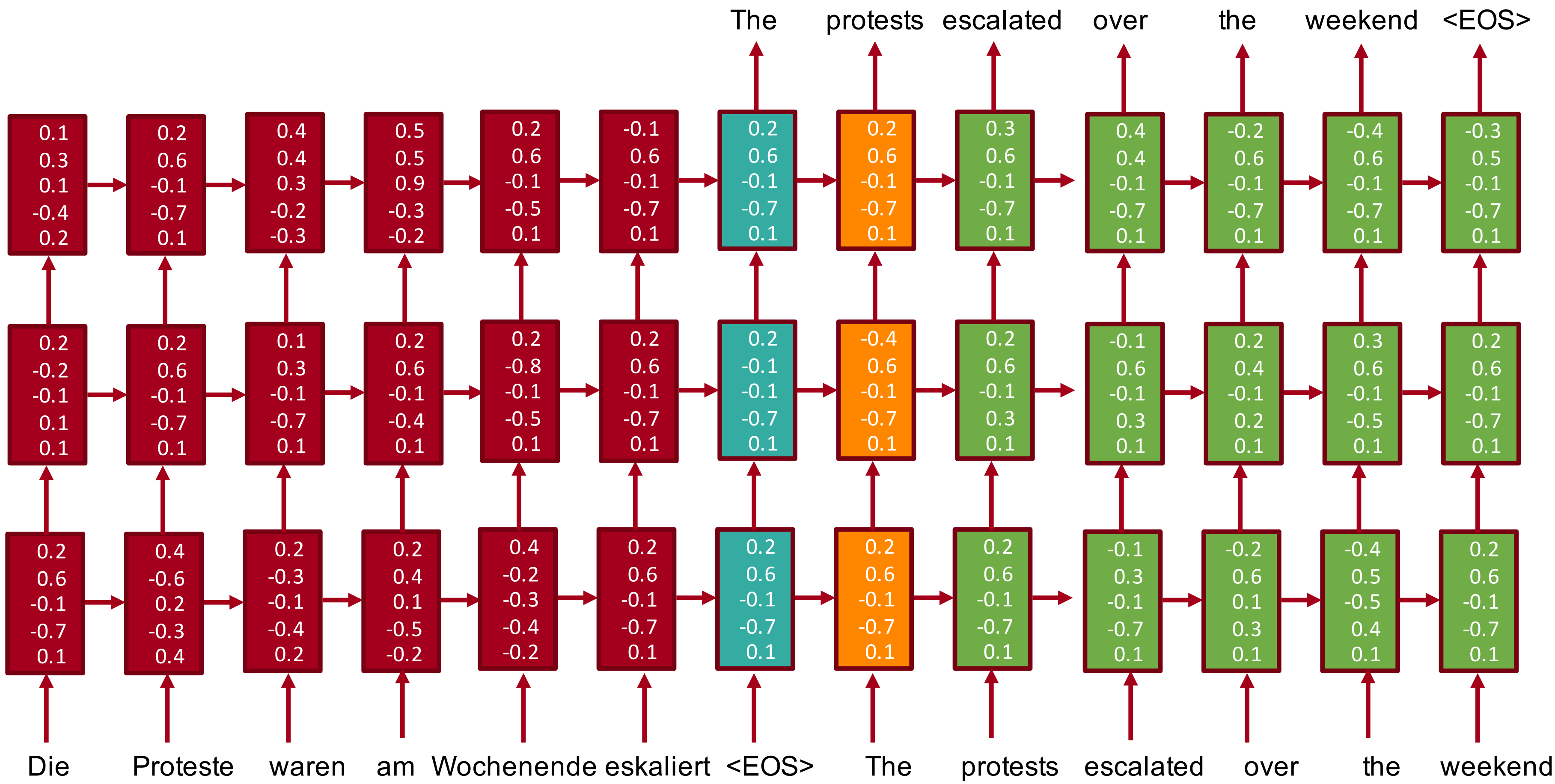
Syntax-based MT: Translation as parsing

[Galley et al.; NAACL 2004]



Neural Machine Translation

[Sutskever, Vinals & Le 2014, Bahdanau, Cho & Bengio 2014, Luong, Pham & Manning 2015]



Lecture Plan

1. Last bits of Machine Translation [15 mins]
2. **Language Models** [15 mins]
3. Tuning and Evaluating Models [5 mins]
4. Final Project Discussion

Language Models

- Traditional grammars (e.g., regular, context free) give a hard ("categorical") model of the sentences in a language
- For NLP, and other applied work, a probabilistic model of a language is *much* more useful
  - It says what people usually say (next)
  - It enables more fine-grained prediction and inference
- Called a Language Model … strange but standard

Watch some videos!

- cs124/nlp-class Language modeling videos, on the OpenEdX site

Do some reading!

- J&M chapter 4
- FSNLP chapter 6
- Chen and Goodman (1998) … for a lot of info

There are may uses of language models … they're NLP's secret weapon

- Speech recognition
  - "I saw a van" is a more likely sentence than "eyes awe of an"
- OCR & Handwriting recognition
- Machine translation
  - More likely sentences are probably better translations
- (Fluent Text) Generation
  - More likely sentences are probably better NL generations
- Context sensitive spelling correction
  - "Their are problems wit this sentence."
- Predictive text input systems
  - Please turn your cell phone of_
- Text classification, gender/style/level detection, information retrieval (LMIR), aspects of grammar checking, text compression, …

Probabilistic Language Models

- Idea is to build models which assign scores to sentences
  - P(I saw a van) >> P(eyes awe of an)
  - Not really grammaticality
    - P(artichokes intimidate zippers) ≈ 0
- Formally, a probability distribution over sentences of a language … sums to 1 over whole language

- Try: empirical distribution over training corpus sentences?
  - Problem: doesn't generalize (at all)
  - Whereas languages are infinite

Probabilistic Language Models

Major components of generalization

- ***Decomposition***: sentences generated in small steps
- ***Discounting***: save some probability mass for the possibility of unseen events
- ***Backoff*** contexts that words are generated from to equivalence classes of contexts which generalize better
- ***Sharing*** or partial sharing of weights between words

After that, there are a lot of details

- But the details are ***very*** important in getting excellent performance in many NLP systems

Decomposition: N-Gram Language Models

- No loss of generality to break sentence probability down with the chain rule

$$P(w_1 w_2 \dots w_n) = \prod_i P(w_i \mid w_1 w_2 \dots w_{i-1})$$

- Too many histories!
  - P(??? | No loss of generality to break sentence) ?
  - P(??? | the water is so transparent that) ?

- N-gram solution: assume each word depends only on a short linear history (a Markov assumption) = equivalence classing

$$P(w_1 w_2 \dots w_n) = \prod_i P(w_i \mid w_{i-k} \dots w_{i-1})$$

$$= \prod_i P(w_i \mid w_{i-1}) \text{ for bigram}$$

Entropy

- Claude Shannon (1951): the entropy of English
  - http://www.math.ucsd.edu/~crypto/java/ENTROPY/

$$H(X) = E_P \log \frac{1}{P(X)}$$

$$= -\sum_{x \in \mathcal{X}} P(x) \log P(x)$$

- Per-word/character cross entropy

$$H(S \mid M) = \frac{-\log_2 P_M(S)}{|S|} = \frac{-\sum_{i=1,\ldots N} \log_2 P_M(w_i \mid w_{1,\ldots,i-1})}{N}$$

e.g.,

$$\sum_j \log_2 P_M(w_j \mid w_{j-1})$$

Word-level entropy

The Palestinian security chief in Gaza **denied the** report

Judge Kathleen Kennedy-Powell **denied the** motion to strike

Pineau-Valencienne has **denied the** charges

The FDA **denied the** group's request

the show's writer and co-star, **denied the** characters had real-life

The district attorney's office had **denied the** KCBS-TV report

Coleman **denied the** charge

Defense attorney Al Kitching **denied the** allegations

Local officials have consistently **denied the** existence of armed

Kraft has categorically **denied the** remarks

Goddard has **denied the** charges

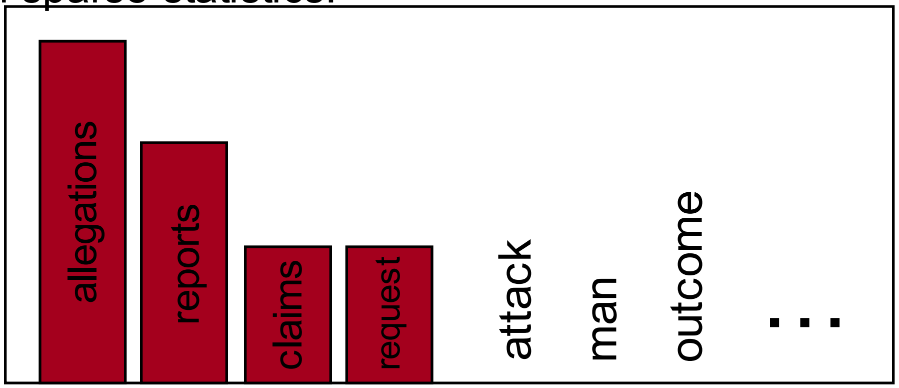
congressional employees are **denied the** legal protections

who **denied the** accusation of the woman

Discounting/Smoothing

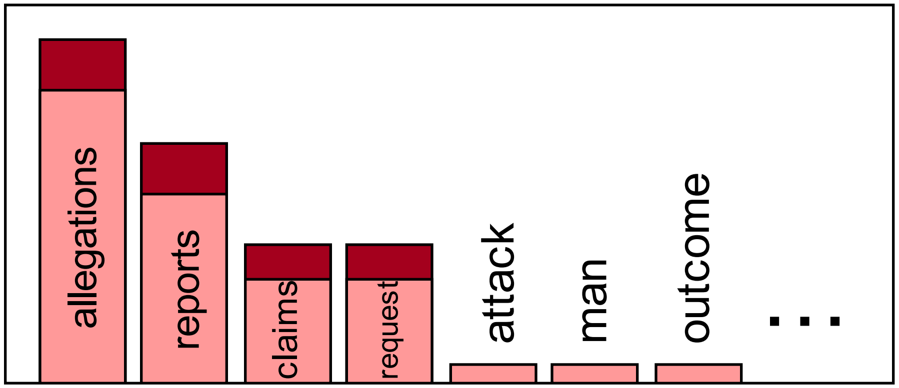
- We often want to make estimates from sparse statistics:

P(w | denied the)
3 allegations
2 reports
1 claims
1 request
7 total



- Smoothing flattens spiky distributions so they generalize better

P(w | denied the)
2.5 allegations
1.5 reports
0.5 claims
0.5 request
2 other
7 total



- Very important all over NLP, but easy to do badly!
- Illustration with trigrams (h = previous two words, could be anything).

Discounting/Backoff/Interpolation

- $P(w_i|h)$ is just a multinomial
    - but we need to estimate it well
  - We want to know how often a word follow some history h
  - There's some true distribution $P(w \mid h)$
  - We saw some small sample of N words from $P(w \mid h)$
  - We want to reconstruct a useful approximation of $P(w \mid h)$
  - Counts of events we didn't see are always too low
  - Counts of events we did see are *in aggregate* too high

- Discounting: providing mass for what we haven't seen
- Backoff: Increasing N by decreasing the amount of history h
- Interpolation between backed-off distributions: how to best average to allocate mass amongst rarely/unseen events

Language models

- Language models are a cool technology
- You can build them for not only a language like “English” but for particular languages/topics
  - Papers about a topic, like “language modeling”
  - As a character-level language detector
  - Genres like “Seventeenth century novels” or “fan fiction”
- Because they flexibly model higher order context, they can be very powerful models
  - And work very well

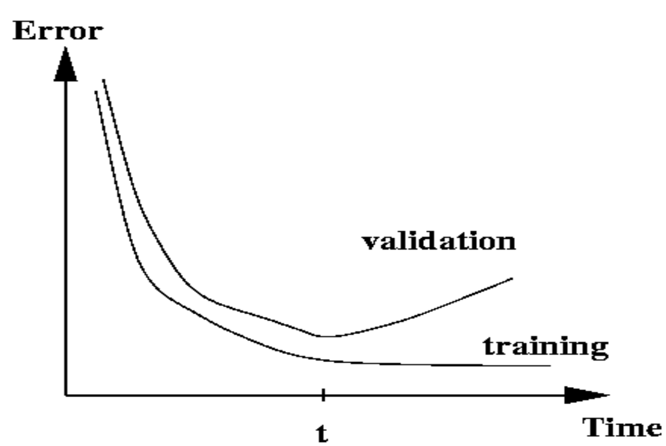
Look at the videos and J&M chapter 4!

Lecture Plan

1. Last bits of Machine Translation [15 mins]
2. Language Models [15 mins]
3. **Tuning and Evaluating Models** [5 mins]
4. Final Project Discussion

Training models and pots of data

- The big danger when training models is that you **overfit** to what you are training on
  - The model correctly describes what happened to occur in particular data you trained on, but the patterns are not general enough patterns to be likely to apply to new data
- The way to monitor and avoid overfitting is using **independent** validation and test sets …



Training models and pots of data

- You build (estimate/train) a model on a **training set**.
- Commonly, you then set further hyperparameters on another, independent set of data, the **tuning set**
  - The tuning set is the training set for the hyperparameters!
- You measure progress as you go on a **dev set** (development test set or validation set)
  - If you do that a lot you overfit to the dev set so it's good to have a second dev set, the **dev2** set
- **Only at the end**, you evaluate and present final numbers on a **test set**
  - Use final test set **extremely** few times … ideally only once

Training models and pots of data

- The **train**, **tune**, **dev**, and **test** sets need to be completely distinct
- It is invalid to test on material you have trained on
  - You will get a falsely good performance. We usually overfit on train
- You need an independent tuning set
  - The hyperparameters won't be set right if tune is same as train
- If you keep running on the same evaluation set, you begin to overfit to that evaluation set
  - Effectively you are "training" on the evaluation set … you are learning things that do and don't work on that particular eval set and using the info
- To get a valid measure of system performance you need another untrained on, **independent** test set … hence dev2 and final test
  - Ideally, you only test on it once … definitely extremely few times

Lecture Plan

1. Last bits of Machine Translation [15 mins]
2. Language Models [15 mins]
3. Tuning and Evaluating Models [5 mins]
4. **Final Project Discussion**