Decoding in SMT

Nitin Madnani
February 8, 2006
The Decoding Problem

• Search

• Inputs:
  • Input string
  • Bunch of statistical models
  • A function to assign score to any translation

• Output:
  • Best scoring translation
Mathematically ...

\[ e = \arg \max_{\hat{e}} S(\hat{e}, f) \]
Mathematically ...

\[ e = \arg \max_{\hat{e}} S(\hat{e}, f) \]

Score
(models, candidate, input string)
Mathematically ...

\[ e = \arg \max_{\hat{e}} S(\hat{e}, f) \]
Mathematically ...

$$e = \text{arg max}_{\hat{e}} S(\hat{e}, f)$$

Search operation

$e$ = search space (all possible translations)

Score (models, candidate, input string)
Mathematically...

\[ e = \arg \max \hat{e} \quad S(\hat{e}, f) \]

"Best" Translation

Search operation

Score
(models, candidate, input string)

search space
(all possible translations)
Mathematically ...

Mathematically, the "Best" Translation is determined by the following equation:

\[ e = \arg \max_{\hat{e}} S(\hat{e}, f) \]

where \( S(\hat{e}, f) \) is the Score function over the search space of all possible translations.

**Search operation**

The search operation involves evaluating the Score function for each candidate translation in the search space.

**Score**

The Score is calculated as a product of probabilities or a sum of weighted models:

- **Models =** \( P(e), P(a|f, e); \) Score = \( P(e) \times P(a|f, e) \)
- **Models =** \( P(e), P(f|e), P(e|f), P(a|f, e), P(e|f) \) etc; Score = \( \exp(\sum w_m m_n) \)

**Examples:**

- Models = \( P(e), P(a|f, e); \) Score = \( P(e) \times P(a|f, e) \)
- Models = \( P(e), P(f|e), P(e|f), P(a|f, e), P(e|f) \) etc; Score = \( \exp(\sum w_m m_n) \)
Decoding is hard
Decoding is hard

- Very simple example

\[ f_1 \quad f_2 \quad f_3 \quad f_4 \quad \ldots \quad f_m \]
Decoding is hard

- Very simple example
- Models: LM, Model 1 (1/1)
Decoding is hard

- Very simple example
- Models: LM, Model 1 (1/1)
- Search space: All possible orderings of $e_1 \ldots e_m$
Decoding is hard

• Very simple example

• Models: LM, Model 1 (1/1)

• Search space: All possible orderings of $e_1 \ldots m$

• Picked by the LM
Decoding is hard

- Very simple example
- Models: LM, Model 1 (1/1)
- Search space: All possible orderings of e₁...m
- Picked by the LM
- \( w(e₁ \rightarrow e₂) = p(e₂ | e₁) \)
Decoding is hard

- Very simple example
- Models: LM, Model 1 (1/1)
- Search space: All possible orderings of $e_1...m$
- Picked by the LM
- $w(e_1 \rightarrow e_2) = p(e_2 | e_1)$
- Look familiar?
Decoding is hard

- Very simple example
- Models: LM, Model 1 (1/1)
- Search space: All possible orderings of e₁...m
- Picked by the LM
- \( w(e₁ \rightarrow e₂) = p(e₂ | e₁) \)
- Look familiar?
- TSP - NP Complete!
Problem characteristics

- Clear-cut optimization problem
  - There is always one right answer
- Inherently Complex
  - Number of ways to order words (LM)
  - Number of ways to cover input words (TM)
- Harder than in SR:
  - No left to right input-output correspondence
Decoding Methods

- Stack-based Decoding
  - Most common
  - Almost all contemporary decoders are stack-based

- Greedy Decoding
  - Faster but more error-prone

- Optimal Decoding
  - Finds the optimal translation
  - Really Really Slow!
Stack-based Decoding

• Originally introduced by Jelinek in SR
• Stores partial translations *(hypotheses)* in a *stack*
• Builds new translations by extending existing hypotheses
• Optimal translation guaranteed if given unlimited stack size and search time
• *Note*: stack does not imply LIFO; actually a (priority) queue
Stack-based Decoding

Hypothesis Stack
(finite size and sorted by cost)
Stack-based Decoding

Hypothesis Stack
(finite size and sorted by cost)

Pop (1)
Stack-based Decoding

Hypothesis Stack
(finite size and sorted by cost)

Pop (1)

Extend by translating every possible word (2)
Stack-based Decoding

Hypothesis Stack (finite size and sorted by cost)

Pop (1)

Extend by translating every possible word (2)

Push (3)
Stack-based Decoding

Hypothesis Stack (finite size and sorted by cost)

1. Pop
2. Extend by translating every possible word
3. Push

Repeat (1)-(3) until a **complete** hypothesis is encountered
Heuristic function

- Hypothesis cost = cost of translation so far
- Problem: Shorter hypotheses will push longer ones out
- Solution: Use translation cost + future cost
- Future cost: What it would cost to complete an hypothesis
- A heuristic provides an estimate of the future cost
- No heuristic can be perfect (no monotonicity)
- Need to find another solution
Multi-stack Decoding

- Use multiple stacks
  - One for each subset of the input words ($2^n$)
  - One for each number of words covered (n)
- Extend the top hypothesis from each stack
- Competition is among similar hypotheses
Other Optimizations

• Beam-based Pruning
  • Relative threshold - prune if $p(h) < \alpha \cdot p(h_{\text{best}})$
  • Histogram - Only keep a certain number of hypotheses, prune the rest
  • Can accidentally prune out a good hypothesis

• Hypothesis Recombination
  • If similar($h_1, h_2$) then keep only the cheaper one
  • Risk-free
Greedy Decoding

- Start with the word-for-word English gloss
- Iterate exhaustively over all alignments one simple operation away
  - Add, substitute, change order etc.
- Pick the one with the highest probability
- Commit the change
- Repeat until no improvement possible
Greedy Decoding

• Pros
  • Much much faster
  • Complexity only scales polynomially with sentence length

• Cons
  • Searches only a very small subspace
  • Cannot find best translation if far from gloss
Optimal Decoding

- Transform decoding problem into a TSP instance
  - Foreign words ~ Cities
  - Translations ~ Hotels in cities
  - Cost ~ Distance
- Solve TSP using Integer Programming (IP)
  - Cast tour selection as a constrained integer program
  - Can find tours of various lengths (n-best lists)
Optimal Decoding

• Pros
  • Fast decoder development
  • Optimal n-best lists
  • Extremely customizable

• Cons
  • Extremely slow!
  • Hard to integrate non-related information sources
Decoding Errors

- **Search Error**
  - \( \text{decode}(f) = e \), but \( \exists e' \) s.t. score\((e')\) > score\((e)\)
  - The right answer is in the space but we couldn’t find it
  - Hard to prove sub-optimal decoding

- **Model Error**
  - \( \text{correct}(f) \notin \text{Search space} \)
  - The right answer is not in the space because of imperfect models
Observations*

• $|\text{space}_{\text{greedy}}| << |\text{space}_{\text{stack}}|$ (hence the speed)

• $\text{space}_{\text{stack}} \subseteq \text{space}_{\text{optimal}}$

• $nSE_{\text{greedy}} >> nSE_{\text{stack}} >> nSE_{\text{optimal}} (=0)$

• $t_{\text{greedy}} < t_{\text{stack}} << < t_{\text{optimal}}$ (50 for $m=6$, 500 for $8!$)

• $nME >> 0$ for all, since Model 4 is deficient

*All decoders are Model 4 and tested on the same set*
Take Home Messages

• Optimal decoding is possible but highly impractical
• Optimized stack-based decoding provides good balance
• All modern decoders are basically the same (stack-based)
  • Differences in models, score and extension operations.  
    Examples: Pharaoh, Rewrite
• Better translations will come from improving models  
  (Hiero)