Part III: Search Problem

- Extensions to translation models and search algorithms
- Re-ordering constraints
- DP beam-search with single and multi-stacks
- Dynamic Programming and the TSP problem
- A* search: with single and multi-stacks
- Complexity issues

Statistical Machine Translation
Decoding

Given a statistical alignment model, a language model, and a source sentence, the task of the search procedure is to find the most likely translation:

\[
\text{Let us consider decoding with Model 1:}
\]

\[
\text{Decoding Complexity}
\]

- Decoding with Model 1 and higher models can be proven to be NP-hard, hence there is little hope for an efficient algorithm.
- Decoding over possible target sentence subsequences requires \(O(g_{w, m})\) operations.
- Even if we assume that any French word might translate into at most \(k\) words, computing the single alignment and LM probabilities is fast.

We assume a fixed range of lengths for the target string, \(e \in \{1, 2, \ldots, 2m\}\).

\[
\begin{align*}
\text{We may use the } & \text{Viterbi or maximum approximation:} \\
\text{the task of the search procedure is to find the most likely translation:} \\
\text{decoding in SMT}
\end{align*}
\]
The problem whether \( dN \neq d \) or \( dN = d \) is still unsolved!

The problem whether \( dN \neq d \) or \( dN = d \) is still unsolved!

- \( dN \neq d \) or \( dN = d \) is still unsolved!

- \( dN \neq d \) or \( dN = d \) is still unsolved!

- \( dN \neq d \) or \( dN = d \) is still unsolved!

- \( dN \neq d \) or \( dN = d \) is still unsolved!

- \( dN \neq d \) or \( dN = d \) is still unsolved!

- \( dN \neq d \) or \( dN = d \) is still unsolved!

- \( dN \neq d \) or \( dN = d \) is still unsolved!

- \( dN \neq d \) or \( dN = d \) is still unsolved!

- \( dN \neq d \) or \( dN = d \) is still unsolved!

- \( dN \neq d \) or \( dN = d \) is still unsolved!

- \( dN \neq d \) or \( dN = d \) is still unsolved!

- \( dN \neq d \) or \( dN = d \) is still unsolved!

- \( dN \neq d \) or \( dN = d \) is still unsolved!

- \( dN \neq d \) or \( dN = d \) is still unsolved!

- \( dN \neq d \) or \( dN = d \) is still unsolved!

- \( dN \neq d \) or \( dN = d \) is still unsolved!

- \( dN \neq d \) or \( dN = d \) is still unsolved!

- \( dN \neq d \) or \( dN = d \) is still unsolved!

- \( dN \neq d \) or \( dN = d \) is still unsolved!

- \( dN \neq d \) or \( dN = d \) is still unsolved!

- \( dN \neq d \) or \( dN = d \) is still unsolved!

- \( dN \neq d \) or \( dN = d \) is still unsolved!

- \( dN \neq d \) or \( dN = d \) is still unsolved!

- \( dN \neq d \) or \( dN = d \) is still unsolved!

- \( dN \neq d \) or \( dN = d \) is still unsolved!

- \( dN \neq d \) or \( dN = d \) is still unsolved!

- \( dN \neq d \) or \( dN = d \) is still unsolved!

- \( dN \neq d \) or \( dN = d \) is still unsolved!

- \( dN \neq d \) or \( dN = d \) is still unsolved!

- \( dN \neq d \) or \( dN = d \) is still unsolved!

- \( dN \neq d \) or \( dN = d \) is still unsolved!

- \( dN \neq d \) or \( dN = d \) is still unsolved!
Decoding with M1 is NP-complete. We have expressed the HCP as a translation problem. As we now that HCP is

insight

Theorem

That $P\text{\textsc{e}}(\text{\textsc{p}}(\text{\textsc{e}} | \text{\textsc{f}})) < \frac{1}{2}$ given a string $e$ of length $\leq 2m$ such that $P_0(e) \cdot P(\text{\textsc{f}} | \text{\textsc{e}})$
and $P_1(e) \cdot P(\text{\textsc{f}} | \text{\textsc{e}})$ there exists a string $e$ of length $\leq 2m$ such that $P_0(e) \cdot P(\text{\textsc{f}} | \text{\textsc{e}})$
and $P_1(e) \cdot P(\text{\textsc{f}} | \text{\textsc{e}})$

M1-DECIDE: We express the search problem in terms of a decision problem
The solution is built left-to-right, but input is consumed in any order.

MT decoding does not progress synchronously with input (as in ASR).

Stack Decoder

• A∗ search needs an heuristic to estimate the completion cost of each theory.

• The set of positions covered by h is called coverage set.

• The set of partial solutions generated by h is called partial coverage set.

• The stack is initialized with an empty hypothesis.

• Return 1.

   - I. Initialize the stack with an empty hypothesis.
   - 2. Pop h, the best hypothesis from the stack.
   - 3. If h is a complete sentence, output h and terminate.
   - 4. Cover some vacant position, possibly extend h with a target word, m, and push the resulting hypothesis onto the stack.

   - 5. Return 2.

   - The stack or A∗ decoding algorithm builds a solution incrementally and stores partial solutions h into an ordered stack.

Progress in Search Algorithms for SMT

Koehn et al.

2003

Och & Ney

2002

Tillmann et al.

2001

Germann et al.

1998

Wang et al.

1997

Wu

1996

Berger et al.

Based SMT (Pharaoh, Moses)

Multistack DP beam-search for phrase-based SMT (Pharaoh, Moses)

Multistack DP beam-search alignment template appr.

DP search alignment template appr.

Greedy, integer programming (short term decoders for Model 4)

Greedy, integer programming (short term decoders for Model 4)

Greedy, integer programming (short term decoders for Model 4)

Greedy, integer programming (short term decoders for Model 4)
MultiStack Decoder

Corresponds to an optimal path covering a specific subset of cities and includes all the information which are necessary to search for a longer path.

\[ \exists (\exists, \mathbb{C}) \]

all cities of the subset \( \mathbb{C} \) must also exist at city \( t \), ending in city \( t \) and visiting cost of the optimal partial tour starting

\[ f_{\text{ap}} (\exists, \mathbb{C}) \]

the recursive quantity

\section{Dynamic Programming Approach}

Let us introduce the Traveling Salesman Problem

Just introduce words with high zero-reliability probability and which increase the score of a theory.

We need some tricks to reduce cost of some operation, e.g. Address: we align a French word to the English NULL word

\begin{itemize}
\item AddNull
\item Extend
\item AddrZer
\item Addz
\end{itemize}

Build solutions incrementally by applying operations to hypotheses (step 4): Idea: use one stack for each coverage set size, extend one \( h \) for each stack
directly comparable

Without a good heuristic hypotheses with different coverage sets are not comparable.
Held and Karp Algorithm for TSP

1. Input: Cities 1, 2, ..., n with distance matrix $d(j, j')$.

2. Initialization: For $k = 2, \ldots, n$, $Q \{ k \} = \emptyset$.

3. For each path length $c = 2, \ldots, n$, $u \cdots u = 1$.

4. For each pair $(f', f)$ with $c = |(f', f)|$:

   $\min_{\text{shortest tour}} (f', f) = (1)$.

5. For each pair $(f', f)$ with $c = |(f', f)|$:

   $\min_{\text{shortest tour}} (f', f) = (1)$.

6. For each pair $(f', f)$ with $c = |(f', f)|$:

   $\min_{\text{shortest tour}} (f', f) = (1)$.

7. For each path length $c = 2, \ldots, n$:

   $u \cdots u = 1$.

8. For each pair $(f', f)$ with $c = |(f', f)|$:

   $\min_{\text{shortest tour}} (f', f) = (1)$.

9. For each pair $(f', f)$ with $c = |(f', f)|$:

   $\min_{\text{shortest tour}} (f', f) = (1)$.

10. For each path length $c = 2, \ldots, n$:

    $u \cdots u = 1$.

11. For each path length $c = 2, \ldots, n$:

    $u \cdots u = 1$.
Held and Karp DP Algorithm for TSP

Each information is necessary to extend the translation:

- \( \mathcal{C} \) is needed to control the coverage of all positions
- \( \mathcal{C} \) is needed to compute the translation probability
- \( \mathcal{C} \) is needed to compute the distortion probability
- \( \mathcal{C} \) is needed to control the coverage of all positions
- \( \mathcal{C} \) is needed to compute the translation probability

The recursive quantity must be enriched with other state information:

- \( \mathcal{C} \) is needed to control the coverage of all positions
- \( \mathcal{C} \) is needed to compute the translation probability
- \( \mathcal{C} \) is needed to control the coverage of all positions
- \( \mathcal{C} \) is needed to compute the translation probability
- \( \mathcal{C} \) is needed to control the coverage of all positions

A DP algorithm for a fertility word-based SMT model can be derived from the

\[ \begin{array}{c}
\text{DP decoder for SMT} \\
\end{array} \]
II. Backtrack solution

11. 10
10. \[ i + 1 = ? \]
9. Prune less promising theories in pool \([i + 1] \]
8. Update backpointer
7. "Put the \([i + 1] \]
6. If \( \theta' \) is partial and improves best theory or its state then
5. If \( \theta' \) is complete and improves current solution then replace solution
4. For all expansions \( \theta' \) of \( \theta \) do
3. Put translation theories covering the null word into \([0] \]
2. Put translation theories covering the null word into \([0] \]
1. \( i = 0 \)

**DP beam-search algorithm**
Basic steps of search:

1. Pick hyp. from one stack
2. Cover new source positions
3. Generate translation options
4. Score hypotheses
5. Recombine hypotheses
6. Put them into stacks
7. Prune stacks

Constraints to reduce expanded hypotheses:

• Beam search: keep at most M hypotheses in the pool; prunes out less promising partial theories
• Lexicon pruning: keep just most probable word translations
• Threshold pruning: keep just theories close to the best theory in the pool

Efficient representation and use of:

• Efficient representation of hypotheses: pruning, quantization, caching

Memory optimization:

• Trie grammar pruning: keep at most M hypotheses in the pool without successors
• Beam search: keep at most M hypotheses in the pool

Solutions for efficiency:

Stack N contains hypotheses covering N positions

Multi-stack DP (Moses)
Extensions of Search Algorithms and Models

- Word-reordering: Max Skip Constraints
  - Word-graphs and N-best lists can be extended to compute confidence scores of translations
  - Re-scored with models that are difficult to integrate in the search algorithm
  - N-best Paths and N-best lists in the word-graph
  - Search algorithms can be extended to compute N-best Paths of translations: dump active hypotheses in search space
    - White state transducer, adaptable to manage word re-ordering
    - Multi-stack decoder, suited to manage word re-ordering

There are alternative implementations of DP beam-search

- Search algorithms can work with different translation models
  - There are alternative implementations of (n-grams)
  - Search algorithms can be extended to compute word-reordering
    - Generalization of word-based translation is phrase-based translation
    - Generalization of the TM-LM model is the so-called Log-linear model
    - Just replace the scoring function of partial hypotheses

- There are alternative implementations of DP beam-search
  - Word-reordering: Max Skip Constraints
    - Example with \( k = 4 \)
  - Complexity of the TSP can be shown to decrease from \( O(n^2) \) to \( O(n \log n) \)

\[
\begin{array}{ccccccccccc}
    & & & & & & & & & & \\
    n & . & . & . & . & . & . & . & . & . & 1 \\
    . & . & . & . & . & 0 & 0 & 0 & 0 & 0 & (p) \\
    . & . & . & . & . & 0 & 0 & 0 & 0 & 0 & (q) \\
    . & . & . & . & . & 0 & 0 & 0 & 0 & 0 & (r) \\
    . & . & . & . & . & 0 & 0 & 0 & 0 & 0 & (s) \\
\end{array}
\]

- Indicate permitted positions for coverage
- Indicate so far covered positions
- Example with \( k = 4 \)
- Constraint: at each step cover one of the first \( k \) empty positions