Statistical Machine Translation: Decoding

Jakub Waszczuk

Heinrich Heine Universität Düsseldorf

Winter Semester 2018/19

Outline

Outline

Decoding: Problem Statement

Given

- Input sentence (sequence of words) $\vec{x}$
- Translation model $P_T(\vec{x} \mid \vec{y})$
- Language model $P_L(\vec{y})$

Goal (theory)

$$\arg\max_{\vec{y}} P_T(\vec{x} \mid \vec{y}) \times P_L(\vec{y})$$

Decoding: Problem Statement

Given

- Input sentence (sequence of words) $\vec{x}$
- Translation model $P_T(\vec{x} \mid \vec{y})$
- Language model $P_L(\vec{y})$

Goal (theory)

$$\arg\max_{\vec{y}} P_T(\vec{x} \mid \vec{y}) \times P_L(\vec{y})$$

Goal (practice)

$$\arg\max_{\vec{y}} \left(\max_{h} P_T(\vec{x}, h \mid \vec{y}) \times P_L(\vec{y}) \right)$$

where h corresponds to some hidden variable (alignment, phrase segmentation, etc.)

Today

Focus

We consider the task of decoding within the context of

- phrase-based translation model
- bigram language model

Decoding

$$\arg\max_{\vec{y}} \left(\max_{\varphi,a} P_T(\vec{x}, \varphi, a \mid \vec{y}) \times P_L(\vec{y}) \right)$$

where

- φ – segmentation of $\vec{x}$ and $\vec{y}$ to phrases
- a – alignment between phrases

Translation as Search

Search problem

Translation can be represented in the form of a *search problem*:

- We have a lot of possible solutions (translations)
- We search for what amounts to be the best solution

Challenge

- The set of possible translations exponential (in general: infinite)
- Infeasible to look at all solutions one by one

Structured Search

Observation

- Translations are structured
- Partial scores can be assigned to partial translations

Translation Process

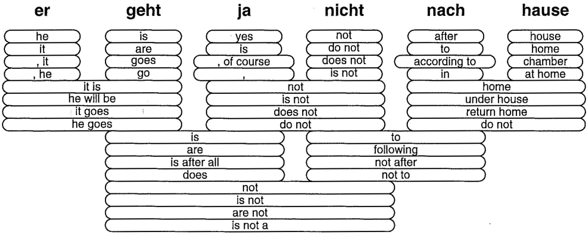
Translating a Sentence

We represent translation as a sequence of steps:

- Start with an empty output sentence
- In each step
  - Select a phrase p in the input sentence
  - Translate p it to an output phrase q
  - Append q at the end of the output translated so far
- Stop when all the words in the input sentence are translated

Translation Process: Example

Phrase translation table

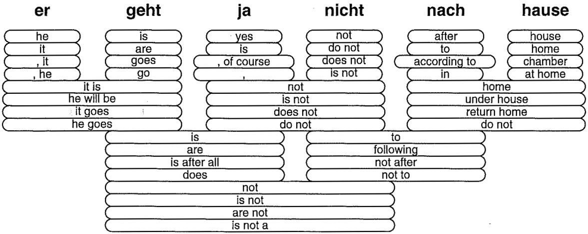


Translation process

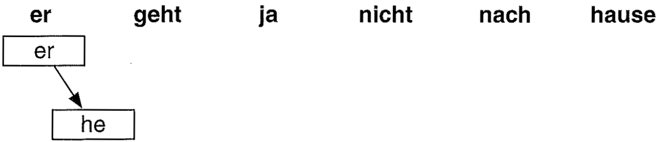
er geht ja nicht nach hause

Translation Process: Example

Phrase translation table

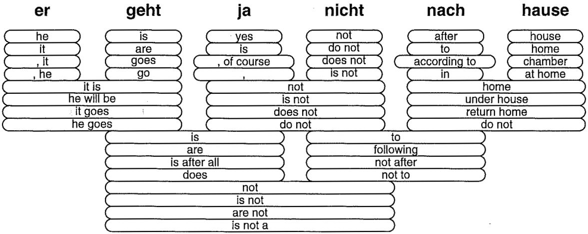


Translation process

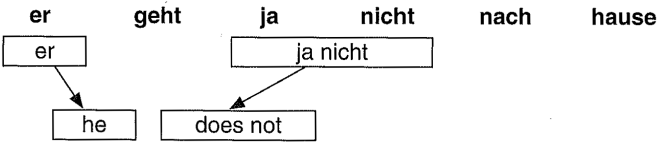


Translation Process: Example

Phrase translation table

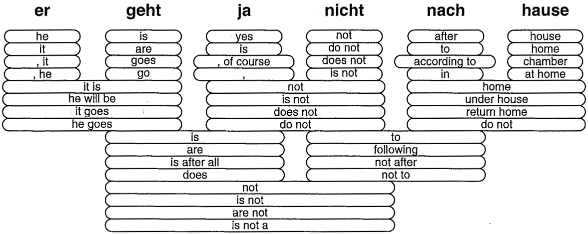


Translation process

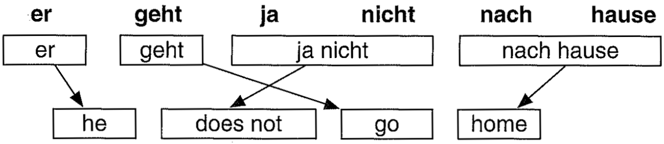


Translation Process: Example

Phrase translation table



Translation process



Scoring

Scoring partial translations

In each translation step, a new phrase gets translated; we factor in:

- The corresponding phrase-translation probability
- The bigram probabilities
- The reordering cost

Scoring

Example

er **geht** **ja** **nicht** **nach** **hause**

Score

$1 \times$

Scoring

Example

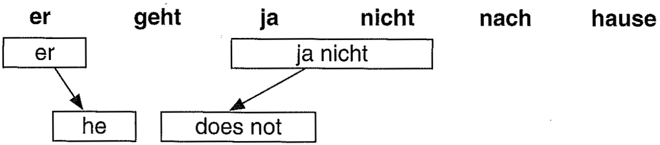
Score

$1 \times$

$P_T(\text{er} \mid \text{he}) \times P_L(\text{he}) \times c(0) \times$

Scoring

Example



Score

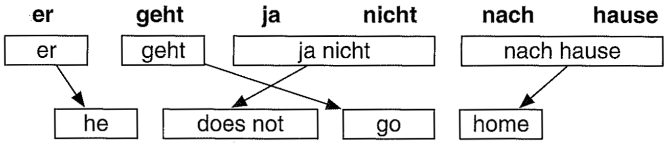
$1 \times$

$P_T(\text{er} \mid \text{he}) \times P_L(\text{he}) \times c(0) \times$

$P_T(\text{ja nicht} \mid \text{does not}) \times P_L(\text{does not} \mid \text{he}) \times c(1) \times$

Scoring

Example

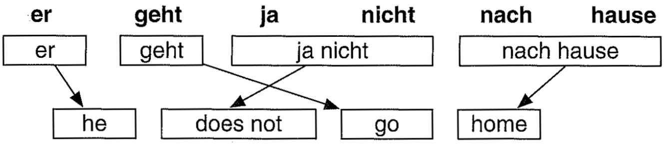


Score

$1 \times$
$P_T(\text{er} \mid \text{he}) \times P_L(\text{he}) \times c(0) \times$
$P_T(\text{ja nicht} \mid \text{does not}) \times P_L(\text{does not} \mid \text{he}) \times c(1) \times$
$P_T(\text{geht} \mid \text{go}) \times P_L(\text{go} \mid \text{not}) \times c(-3) \times$

Scoring

Example



Score

$1 \times$
$P_T(\text{er} \mid \text{he}) \times P_L(\text{he}) \times c(0) \times$
$P_T(\text{ja nicht} \mid \text{does not}) \times P_L(\text{does not} \mid \text{he}) \times c(1) \times$
$P_T(\text{geht} \mid \text{go}) \times P_L(\text{go} \mid \text{not}) \times c(-3) \times$
$P_T(\text{nach hause} \mid \text{home}) \times P_L(\text{home} \mid \text{go}) \times c(2)$

Translation Process

Non-determinism

At any given step of the translation process

- There are many input phrases to choose from
- Each input phrase can be translated to several output phrases

The process of translation is non-deterministic

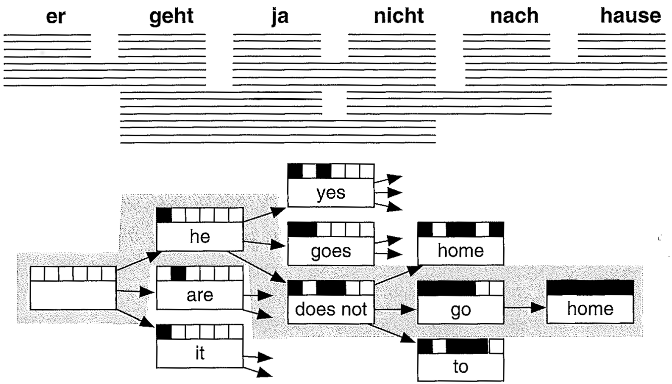
Tree Search

Idea

- Represent the translation process in the form of a *search tree*
- Each branch in this tree (path from the root to a leaf) represents a translation process
- We say that a leaf is *complete* if it represents a complete translation
- Goal: determine the highest-scoring complete leaf

Tree Search

Example



Tree Search

Formalization

We now formalize the process of construction of the search tree. We need:

- **Hypothesis**: node in the search tree / formal representations of a partial translation
- **Expansion**: arc in the search tree / process of determing next translation step
- **Exploration**: algorithm for tree traversal

Hypothesis

Definition

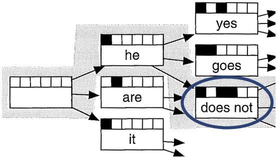
Let $x_1 \dots x_n$ be the input sentence of length n.

A *hypothesis* is a 4-tuple $h = \langle M, p, e, w \rangle$ where:

- $M \subseteq \{1, \dots, n\}$ is the set of input positions translated so far
- p is the last output phrase of the partial translation generated so far[a]
- e is the last input position of the last translated phrase
- w is the partial weight/score of the generated partial translation

[a] This is enough in case of the bigram model.

Example (*er geht ja nicht nach hause*)



- $M = \{1, 3, 4\}, \quad p = \text{does not}, \quad e = 4$
- $w = P_T(\text{ja nicht} \mid \text{does not}) \times P_T(\text{er} \mid \text{he}) \times P_L(\text{he does not}) \times c(0) \times c(1)$

Hypothesis

Definition

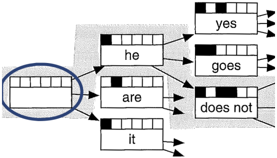
Let $x_1 \ldots x_n$ be the input sentence of length n.

A *hypothesis* is a 4-tuple $h = \langle M, p, e, w \rangle$ where:

- $M \subseteq \{1, \ldots, n\}$ is the set of input positions translated so far
- p is the last output phrase of the partial translation generated so far[a]
- e is the last input position of the last translated phrase
- w is the partial weight/score of the generated partial translation

[a] This is enough in case of the bigram model.

Example (*er geht ja nicht nach hause*)



- $M = \emptyset, \quad p = \rtimes, \quad e = -1$
- $w = 1$

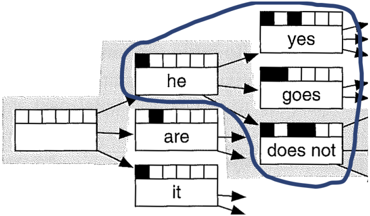
Hypothesis Expansion

Expansion

Given hypothesis *h*, list all hypothesis which expand on *h* by

- selecting a not-yet-translated contiguous fragment in the input sentence
- selecing a possible translation of this fragment according to the phrase translation table
- creating the hypothesis resulting from the selected phrase translation

Example



Hypothesis Expansion

Algorithm 1 Hypothesis expansion (simple, could be optimized)

```
given hypothesis h = ⟨M, p⃗, e, w⟩
for i = 1 ... n do
    for j = i ... n do
        if {i, i + 1, ..., j} ∩ M = ∅ then        ▹ the selected span must not be translated yet
            let x⃗ = x_i ... x_j                    ▹ input phrase to translate
            for each y⃗ ∈ R_E(x⃗) do                ▹ for each potential translation of x⃗
                let w′ = w × P_T(x⃗ | y⃗) × P_L(y⃗ | p⃗) × c(|i − e − 1|)
                let M′ = M ∪ {i, i + 1, ..., j}
                w′ ← w′ × P_L(⋊ | y⃗) if M′ = {1, 2, ..., n}   ▹ in case of complete hypothesis
                yield ⟨M′, y⃗, j, w′⟩
            end for
        end if
    end for
end for
```

Search Tree Exploration

Exploration algorithms

Different algorithms, with different trade-offs, can be used to explore the search tree:

- Depth-first search
- Breadth-first search
- ...

Search Tree Exploration

Algorithm 2 Breadth-first search

```
let Q be an empty queue of hypotheses
let G be an empty set of completed hypotheses
place empty hypothesis in Q
while Q not empty do
    remove h from Q
    if h complete then
        add h to G
    else
        for each expansion h′ of h do
            add h′ to Q
        end for
    end if
end while
```

Search Tree Exploration

Issue

Standard graph-exploration algorithms (such as breadth-first search) are impractical (except for very short sentences), because:

- The first solution found is not enough (**why?**)
- The entire search tree is explored
- The size of this tree is exponential

Outline

Optimized Search

Idea

- Focus on the *promising* parts of the search tree
- We need to be able to answer the following question:

  given two nodes v and w, which of them is more promising to explore?
- Promising $\equiv$ with higher scores

Stack Decoding

Comparable hypotheses

- We say that two hypothesis $h = \langle M, p, e, w \rangle$ and $h' = \langle M', p', e', w' \rangle$ are *comparable* if

$$|M| = |M'| \tag{1}$$

- Idea: the scores of comparable hypotheses involve roughly the same number of multiplications – hence, they can be meaningfully compared

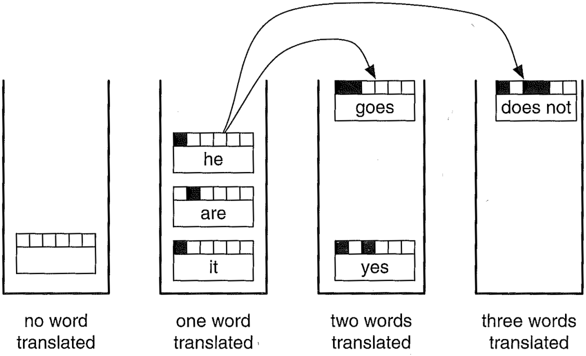
Stack Decoding

Idea

- Hypothesis are organized into groups, called *stacks*, with hypothesis present in the same stack being comparable between each other
- We start with the empty hypothesis, as in tree search
- The subsequent stacks are gradually filled via hypothesis expansion

Stack Decoding

Example (*er geht ja nicht nach hause*)



Stack Decoding

Algorithm 3 Pseudocode

place empty hypothesis into stack 0
for each stack $i = 0 \ldots n-1$ **do**
    for each hypothesis h in stack i **do**
        for each expansion h' of h **do**
            let k be the number of translated words in h'
            place h' in stack k
        end for
    end for
end for

Stack Decoding

Properties

- Stack decoding provides a different startegy of exploring the space of hypothesis
- Computationally, it still involves generating all possible hypothesis and translations
- It's advantage lies in the fact that it allows convenient pruning heuristics

Outline

Optimization Strategies

Pruning

- Idea: trim the branches considered as not promising/useless based on partial scores
- Examples: dead-end detection (exact), branch-and-bound (exact), **hypothesis recombination** (exact), **beam search** (approximate)

Score-guided exploration

- Idea: explore the nodes of the search graph in an order consistent with the scores
- Goals: (i) find the optimal (or close to optimal) hypothesis, (ii) explore as small a part of the search graph as possible
- Examples: shortest-path algorithms (Dijkstra, $A^\star$)

Note: we will look more closely at the techniques marked in bold

Hypothesis Recombination

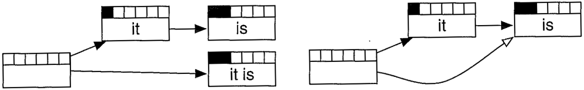
Recombination

Let $h = \langle M, p, e, w \rangle$ and $h' = \langle M', p', e', w' \rangle$ be two hypothesis. Let also $last(x)$ be the last word of phrase x. Then, if:

- $M = M'$
- $last(p) = last(p')$
- $w > w'$

we can safely ignore (prune) h' and all its direct and indirect expansions.

Example



Hypothesis Recombination

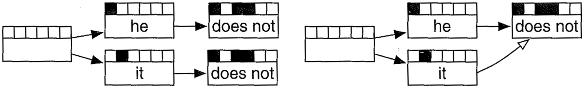
Recombination

Let $h = \langle M, p, e, w \rangle$ and $h' = \langle M', p', e', w' \rangle$ be two hypothesis. Let also $last(x)$ be the last word of phrase x. Then, if:

- $M = M'$
- $last(p) = last(p')$
- $w > w'$

we can safely ignore (prune) h' and all its direct and indirect expansions.

Example



Hypotheses Recombination

Algorithm 4 Stack Decoding with hypotheses recombination

```
place empty hypothesis into stack 0
for each stack i = 0 ... n − 1 do
    for each hypothesis h in stack i do
        for each expansion h′ of h do
            let k be the number of translated words in h′
            place h′ in stack k
            recombine h′ with another hypothesis in stack k if possible
        end for
    end for
end for
```

Hypotheses Recombination

Consequences

- Significantly reduced search space
- Still, in practice, not enough for efficient decoding

Outline

Beam Search

Idea

Beam search combines stack decoding with

- histogram pruning
- threshold pruning

Beam Search

Histogram pruning

Given parameter $K > 0$

- limit each stack to the K hypotheses with the best scores

Beam Search

Histogram pruning

Given parameter $K > 0$

- limit each stack to the K hypotheses with the best scores

Threshold pruning

Given parameter α, for any stack k

- let w_{best} be the best score in stack k
- remove from stack k any hypothesis with score smaller than $\alpha \times w_{best}$

Beam Search

Algorithm 5 Stack decoding with pruning

place empty hypothesis into stack 0
for each stack $i = 0 \ldots n - 1$ **do**
  for each hypothesis h in stack i **do**
    for each expansion h' of h **do**
      let k be the number of translated words in h'
      place h' in stack k
      recombine h' with another hypothesis in stack k **if** possible
      <u>prune stack k **if** necessary</u>
    end for
  end for
end for

Beam Search

Consequences

- Histrogram pruning guarantees efficient (polynomial) decoding
- Threshold pruning ,,smarter” but no efficiency guarantees
- In practice, combination of both techniques typically used

Optimization

Other optimization techniques used in SMT

- Remaining score estimation
- Shortest-path $A^\star$ algorithm

More on them in the complementary material