Machine Translation - Decoding

Christian Kranzler, Hannes Pomberger

January 15, 2007

Table of Contents

IBM Model 4 (Recap)

- Word alignments
- Fertility Table
- Translation Table
- Heads
- Non-heads
- NULL-generated

IBM Model 4 (Recap) (ct.)

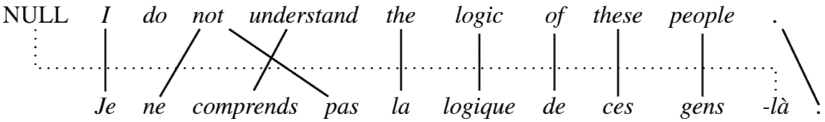


Figure: A sample word alignment [1]

Definition of the search problem

- Sum of the probabilities of aligning f with e
- $P(f \mid e) = \sum_a P(a, f \mid e)$
- Calculating this sum is very expensive, therefore:
- Maximize term $P(a, f \mid e) * P(e)$

Definition of the search problem (ct.)

- Definition of the search problem:

$$< \hat{e}, \hat{a} >= argmax_{e,a} P(a, f \mid e) * P(e)$$

The basic algorithm

- Best-First search algorithm
- Very similar to the A* algorithm
- Hypotheses stored in a priority queue

The basic algorithm (ct.)

- (1) Initialize the stack with an empty hypothesis
- (2) Pop *h*, the most promising hypothesis, off the stack
- (3) If *h* is complete, output *h* and terminate
- (4) Extend *h* in each possible manner of incorporating the next input word and insert the resulting hypotheses into the stack
- (5) Return to step (2)

Stack decoding: machine translation versus speech recognition

- Speech recognition follows input order
- $\rightarrow$ decoder can take words in any order, that increases complexity up to $n!$ permutations of an n-word input
- Heuristic function estimates cost of completing partial hypotheses

Stack decoding - operations

- Add
- AddZFert
- Extend
- AddNull

Stack decoding - pros and cons

- Advantages
  - Explores larger search space than greedy decoder
  - Runs faster than the optimal (Integer Programming) decoder
  - Can employ trigrams (optimal decoder only bigrams)
- Disadvantages
  - Time and space complexity exponential to input words
  - In practise only useful for sentences with max. 20 words

The greedy decoder - the idea

- Time for optimal decoding increases exponentially with input length
- With greedy methods time reduces to polynomial function
- →Start with a random, approximate solution
- →Try to improve it incrementally until satisfactory solution is reached

The greedy decoder - operations

- translateOneOrTwoWords($j, e'_{a(j)}, k, e'_{a(k)}$)
- translateAndInsert($j, e'_{a(j)}, e'_x$)
- removeWordOfFertility0(i)
- swapSegments(i_1, i_2, j_1, j_2)
- joinWords(i, j)

Decoding example

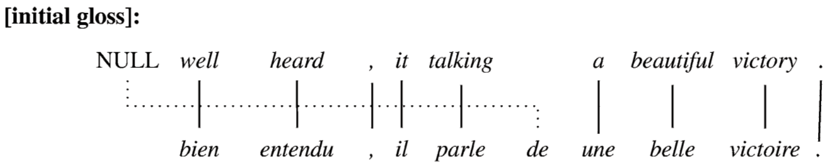


Figure: Initial gloss [1]

Decoding example (ct.)

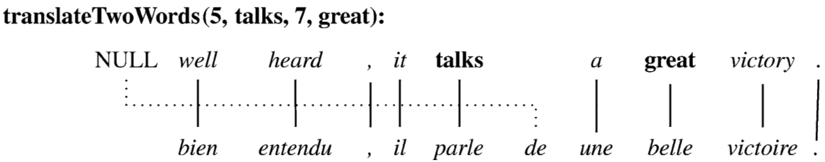


Figure: Step 1 [1]

Decoding example (ct.)

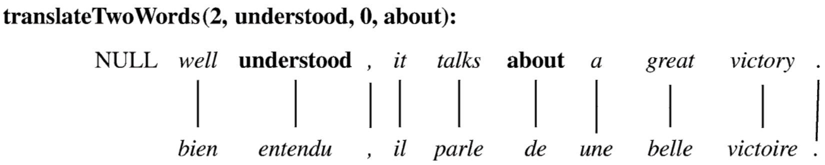


Figure: Step 2 [1]

Decoding example (ct.)

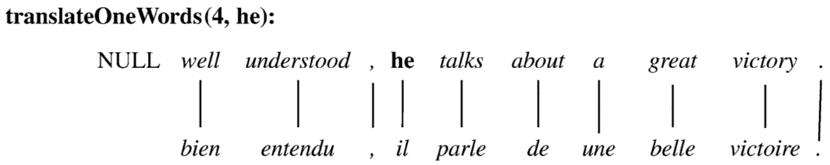


Figure: Step 3 [1]

Decoding example (ct.)

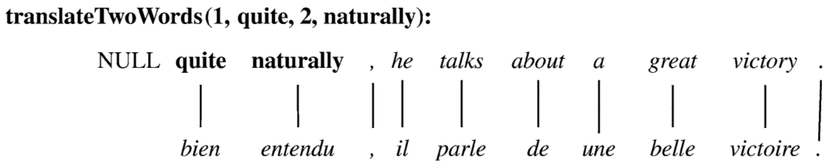


Figure: Step 4 [1]

Integer programming decoding

Idea

- good word order for a decoder output similar to a good TSP tour
- possible to transform a decoding problem into a TSP instance?
- take advantage of previous research into TSP algorithms

Integer programming decoding

- convert decoding into straight TSP is difficult
- more general framework for optimization problems:

  linear integer programming

Integer programming decoding

Sample integer program:

```
minimize objective function:
  3.2 * x1 + 4.7 * x2 - 2.1 * x3
subject to constraints:
  x1 - 2.6 * x3 > 5
  7.3 * x2 > 7
```

Optimal solution:

- respects the constraints
- minimizes the value of the objective function

Integer programming decoding

How to express MT decoding in IP format:

1. create a salesman graph
2. establish real-valued distances
3. cast tour selection as an integer program
4. invoke a IP solver

Integer programming decoding

Create a sales man graph:

- a city for each word in the sentence f
- ten hotels per city corresponding to ten likely English word translations
- if n cities have hotels owned by the same owner x, build $2^n - n - 1$ new hotels on various city boarders
- add an extra city representing the sentence boundary

Integer programming decoding

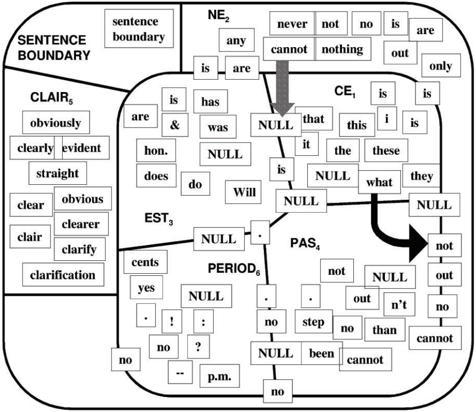


Fig. 3. A **salesman graph** for the input sentence **f** = "*CE NE EST PAS CLAIR.*" A few hotels are omitted for readability, e.g., the *NULL* hotel at the intersection of *EST*, *PAS*, and *PERIOD*.

Integer programming decoding

Define a tour of cities:

- sequence of hotels so that each city is visited once
- hotels on the border of two cities count as visiting both cities
- each tour corresponds to a potential decoding $\langle e, a \rangle$ (owners of the hotel give e, hotel locations yield a)

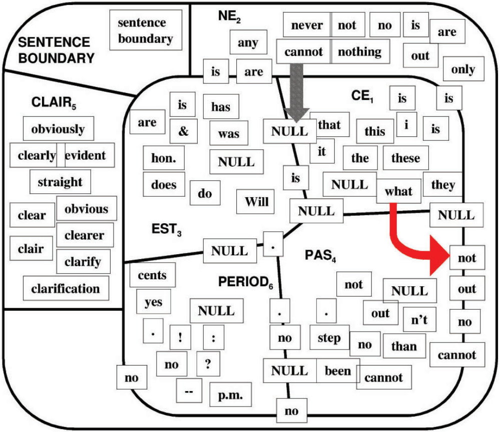
Integer programming decoding

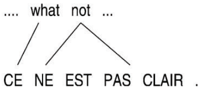
Establish real-valued (asymmetric) distances between pairs of hotels:

- so that length of any tour is $-log(P(e) \cdot P(a, f|e))$
- the distance of each pair of hotels is a piece of the Model 4 formula

Integer programming decoding

Example: inter-hotel distance "not" following "what"





Integer programming decoding

Example: inter-hotel distance "not" following "what"

$$\begin{aligned} distance = \quad & -\log\big(bi(not|what)\big) - \log\big(n(2|not)\big) \\ & -\log\big(t(NE|not)\big) - \log\big(t(PAS|not)\big) \\ & -\log\Big(d_1\big(+1|class(what), class(NE)\big)\Big) \\ & -\log\Big(d_{>1}\big(+2|class(PASS)\big)\Big) \end{aligned}$$

Integer programming decoding

Special treatment of NULL-owned hotels:

- all non-NULL hotels be visited before any NULL hotels
- at most one NULL hotel be visited on a tour

- zero distance from NULL hotel to the sentence boundary hotel
- infinite distance to any other

Integer programming decoding

Example: inter-hotel distance "NULL" following "cannot"

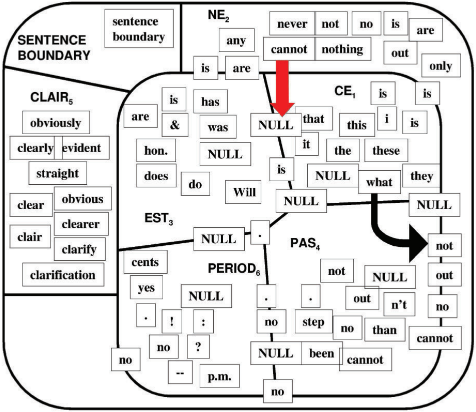


Fig. 3. A **salesman graph** for the input sentence **f** = "*CE NE EST PAS CLAIR.*" A few hotels are omitted for

Integer programming decoding

Example: inter-hotel distance "NULL" following "cannot"

$$\begin{aligned} distance = \quad & -\log \binom{6-2}{2} - 2 \cdot \log(p_1) - (6-4)\log(1-p_1) \\ & -\log\big(t(CE|NULL)\big) - \log\big(t(ES|NULL)\big) \\ & -\log\big(bi(sentence - boundary|cannot)\big) \end{aligned}$$

Integer programming decoding

- between hotels located in the same city assign infinite distance
- for a 6-word French sentence - graph with 80 hotels and 3500 finite-cost travel segments
- Zero-fertility words:
  - disallow adjacent zero-fertility words
  - compare bigram and $n(0|e)$ probabilities

Integer programming decoding

Cast tour selection as an integer program

- sub-tour elimination strategy like used in standard TSP
- create a binary integer variable x_{ij}
- $x_{ij} = 1$ iff travel from hotel i to hotel j

Integer programming decoding

- objective function:

$$\textit{minimize:} \sum x_{ij} \cdot \textit{distance}(i, j)$$

- constraints:
  - exactly one tour segment must exist in each city
  - the segments must be linked to one another
  - prevent multiple independent sub-tours

Integer programming decoding

The shortest tour corresponds to optimal decoding

- invoke a IP solver
  (generic problem solving software such as *lp_solve* or *CPLEX*)
- extract $\langle e, a \rangle$ from the list of variables and their binary values

Integer programming decoding

Further opportunities:

- create a list of n-best solutions
- new constraint to the IP - don't choose the same solution again
- simply repeating the procedure

Integer programming decoding

Advantages of the IP approach in general

+ a decoder can be built rapidly, with very little programming
+ optimal *n*-best results can be obtained
+ generic problem solvers offer a wide range of user-customizable search strategies, thresholds, ect.

Integer programming decoding

Disadvantages

- other knowledge sources (e.g. wider English context) may not be easily integrated
- slow performance

Experiments and discussion

Experiment specification:

- top ten word translations
- 128 words of fertility 0
- bigram language model (Table 1)
- trigram language model (Table 2)
- 505 sentences, uniformly distributed across the lengths 6,8,10,15 and 20

Experiments and discussion

Error classes:

Error classification	$e' = \hat{e}$	$\hat{e}$ is perfect	e' is perfect
1. no error (NE)	yes	yes	yes
2. pure model error (PME)	yes	no	no
3. deadly search error (DSE)	no	yes	no
4. fortuitous search error (FSE)	no	no	yes
5. harmless search error (HSE)	no	yes	yes
6. compound error (CE)	no	no	no

$\hat{e}$ . . . optimal decoding
e' . . . best decoding found by a decoder

Experiments and discussion

Table 1
Comparison of decoders on sets of 101 test sentences. All experiments in this table use a bigram language model. Translation errors can be syntactic, semantic, or both. Errors were counted on the sentence level, so that every sentence can have at most one error in each category

len	decoder	time	SE	TE	NE	PME	DSE	FSE	HSE	CE	BLEU
6	IP	47.50	0	57	44	57	0	0	0	0	0.206
6	stack	0.79	5	58	43	53	1	0	0	4	0.209
6	greedy	0.07	18	60	38	45	5	2	1	10	0.197
8	IP	499.00	0	76	27	74	0	0	0	0	0.157
8	stack	5.67	20	75	24	57	1	2	2	15	0.162
8	greedy	2.66	43	75	20	38	4	5	1	33	0.147

len: input sentence length; **time:** average translation time (in sec./sent.); **SE:** search errors; **TE:** translation errors; **NE:** no error; **PME:** pure model errors; **DSE:** deadly search errors; **FSE:** fortuitous search errors; **HSE:** harmless search errors; **CE:** compound error; **BLEU:** score according to the IBM BLEU metric.

Experiments and discusion

Table 2
Comparison between decoders using a trigram language model. Greedy* and greedy_1 are greedy decoders optimized for speed

Length	Decoder	Av. time (in sec./sent.)	Erroneous translations	BLEU
6	stack	13.72	42	0.282
6	greedy	1.58	46	0.226
6	greedy*	0.07	46	0.202
8	stack	45.45	59	0.231
8	greedy	2.75	68	0.188
8	greedy*	0.15	69	0.174
10	stack	105.15	57	0.271
10	greedy	3.83	63	0.247
10	greedy*	0.20	68	0.225
15	stack	>2000	74	0.225
15	greedy	12.06	75	0.202
15	greedy*	1.11	75	0.190
15	greedy_1	0.63	76	0.189
20	greedy	49.23	86	0.219
20	greedy*	11.34	93	0.217
20	greedy_1	0.94	93	0.209

Experiments and discussion

Conclusions:

- search errors differ significantly between decoders
- measure of translation quality do not

- majority of the translation errors comes from the LM and TM
- for improvement in translation quality better models are needed

Experiments and discussion

Conclusions (ct.):

- BLEU score reflects the rank order
  but is a "ballpark figure" estimate of the decoder performance
- depending on the application
  - slow decoder that provides optimal results
  - or fast, greedy decoder with non-optimal but acceptable results

[1] Germann et al.
Fast and optimal decoding for machine translation
Elsevier B.V. 2003.

[2] Yamada, Knight
A decoder for syntax-based statistical MT
Computational Linguistics Conference 2003.

[3] Brown et al.
The mathematics of statistical machine translation: Parameter estimation
Unpublished 1999.

[4] Knight
Decoding complexity in word-replacement translation models
Computational Linguistics Conference 1999.

Thank you for your attention!

-

Feel free to ask quesitons!