Structured language modeling

Ciprian Chelba and Frederick Jelinek

Computer Speech and Language (2000) 14, 283-332

Markus Koskela

November 8, 2001

Contents

- Structured language model
- SLM experiments
- Lattice rescoring
- Speech recognition experiments
- Conclusions

Structured language model

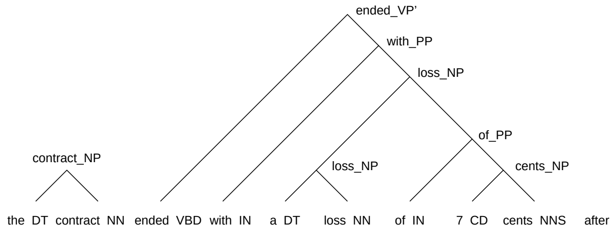
- The n-gram model is far from accurate from a linguistic point of view
- Structured language model (SLM) generalizes trigram by taking advantage of sentence structure
- SLM uses the hierarchical structure of word strings to expand the memory length of prediction
- For efficiency, the model operates left-to-right: $P(w_i|w_1, \ldots, w_{i-1})$
- Model complexity must be limited due to data sparseness
- SLM assigns joint probability to word sequences and hidden parse structure: $P(W_i, T_i)$

Exploiting syntactic structure

- Consider predicting `after` in the sentence:

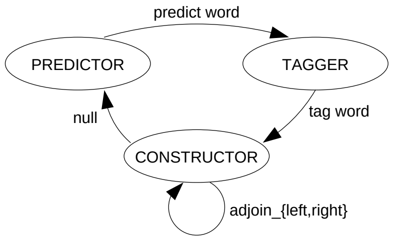
  `the contract ended with a loss of 7 cents | after`

- Trigram predicts `after` from (`7`, `cents`), intuitively it would be better to use (`contract`, `ended`)
- Fix: parse the sentence prefix and use *exposed headwords*:



- Headword percolation and parse tree binarization

Model operation



- *predictor* generates the new word w_k
- *tagger* attaches tag t_k to w_k
- *constructor* builds the partial parse T_k from T_{k-1}, w_k, and t_k in a series of moves ending with a null transition
- A *complete parse* is a parse of (<s>, SB)$(w_1, t_1) \dots (w_n, t_n)$(</s>, SE) so that (</s>, TOP) is the head and $(w_1, t_1) \dots (w_n, t_n)$(</s>, SE) is headed by (</s>, TOP')
- Any given word sequence with a complete parse and headword annotation is generated by a unique sequence of model actions
- Standard trigram belongs to the parameter space of the model

Probabilistic model

- There are multiple parses for a word prefix – need to weight them according to how likely they are

$$P(W,T) =$$
$$\prod_{k=1}^{n+1} P(w_k|W_{k-1}T_{k-1}) P(t_k|W_{k-1}T_{k-1}, w_k) P(T_{k-1}^k|W_{k-1}T_{k-1}, w_k, t_k)$$

($W_{k-1}T_{k-1}$ is the word-parse $(k-1)$-prefix and $T_k = T_{k-1} \,||\, T_{k-1}^k$)

- Only two most recent exposed heads are used:

$$\begin{aligned} P(w_k|W_{k-1}T_{k-1}) &= P(w_k|h_0, h_{-1}) \\ P(t_k|W_{k-1}T_{k-1}, w_k) &= P(t_k|h_0.\mathrm{tag}, h_{-1}.\mathrm{tag}, w_k) \\ P(p_i^k|W_kT_k) &= P(p_i^k|h_0, h_{-1}) \end{aligned}$$

(p_i^k is the ith CONSTRUCTOR action at position k)

Modeling tool

- Model components are conditional probabilistic models
- Probability estimation method is recursive linear interpolation among relative frequency estimates of differerent orders

$$\begin{aligned} P_n(u|z_1, \dots, z_n) &= \lambda(z_1, \dots, z_n) P_{n-1}(u|z_1, \dots, z_n) + \\ &\quad (1 - \lambda(z_1, \dots, z_n)) f_n(u|z_1, \dots, z_n) \\ P_{-1}(u) &= \mathrm{uniform}(\mathcal{U}) \end{aligned}$$

- Interpolation weights are grouped into equivalence classes based on the count $C(z_1, \dots, z_k)$ and determined using *deleted interpolation*:
  - training data is split in "development" and "cross-validation"
  - relative frequency estimates f_k from "development" data
  - EM algorithm for determining the estimates of interpolation weights $\lambda(C(z_1, \dots, z_k))$ from "cross-validation" data

Pruning strategy

- All binary parses with every possible headword annotations are possible due to smoothing $\Rightarrow |\{T_k\}| = \Omega(2^k)$
- Most parses must therefore be pruned, without discarding the most-likely ones
- Syncronous multi-stack pruning algorithm:
  - hypotheses are ranked according to $\log(P(W_k, T_k))$
  - each stack contains partial parses constructed by the same number of parser operations
- Two parameters:
  - maximum number of stack entries (`maximum-stack-depth`)
  - log-probability threshold (`LnP-threshold`)

Parameter estimation

1. Initialization of model parameters
   - each model component is initialized from a (split) treebank after headword percolation and binarization
2. N-best variant of EM re-estimation
   - re-estimation of model parameters to decrease perplexity of the training data
   - modified EM algorithm
3. Last-stage re-estimation
   - separate L2R-WORD-PREDICTOR as the next-word predictor probability need not be the same as the predictor probability used to extract the structure T_k

SLM experiments

- UPenn Treebank corpus, manually annotated data
- word vocabulary 10k, development set 900kwds, check set 74kwds, test data 82kwds
- Pruning parameters: `maximum-stack-depth`=10, `LnP-threshold`=6.91
- Perplexity results:

Iteration number	DEVEL	TEST(λ=0.0)	TEST(λ=0.36)	TEST(λ=1.0)
E0	24.70	167.47	152.25	167.14
E1	22.34	160.76	-	167.14
E2	21.69	158.97	-	167.14
E3=L2R0	21.26	158.28	148.90	167.14
L2R5	17.44	153.76	147.70	167.14

- Iterations are time-consuming, 60h of CPU time on Sun Sparc Ultra-2
- Expected depth of SLM was 3.3–3.4 words (3-gram: 2 words).

Lattice rescoring

- Two-pass recognizer strategies: N-best rescoring and lattice rescoring
- If the two language models have different kind of errors, the weaker model may constrain the stronger too much, not allowing it to show its merits
- Lattice rescoring enables us to have a more complete sample of word hypotheses at a manageable size
- $A*$ algorithm for lattice decoding

Speech recognition experiments

- Previous experiments rather artificial compared to speech recognition:
  - human annotated treebank
  - small amount of training data
  - different word level tokenization
- Training of an parser to generate automatic treebank $\Rightarrow$ more data for SLM training
- Two data sets: WSJ and Switchboard
- WER performance of SLM using the $A*$ lattice decoder
- SLM pruning parameters: `maximum-stack-depth`=10, `LnP-threshold`=6.91

WSJ lattice decoding results

- Read grammatical text in ideal acoustic conditions (40 Mwds)
- Lattices obtained using the standard 3-gram model from DARPA (LAT3-gram), "oracle" WER is 3.4%
- Interpolation with 3-gram is beneficial:

$$P(l) = \lambda P_{\text{3GRAM}}(l) + (1-\lambda) P_{\text{SLM}}(l)$$

- Results:

Decoder	$A*$						Viterbi
λ	0.0	0.2	0.4	0.6	0.8	1.0	1.0
WER(%) (E0 SLM)	14.5	13.1	**12.9**	13.1	13.3	13.7	13.7
WER(%) (E3 SLM)	14.2	13.3	13.2	13.1	13.1	13.7	13.7

Switchboard lattice decoding results

- Conversational speech over a telephone line (2.29 Mwds)
- Lattices obtained using a LAT3-gram language model
- Results:

Decoder	$A*$						Viterbi
λ	0.0	0.2	0.4	0.6	0.8	1.0	1.0
WER(%) (E0 SLM)	42.0	40.8	40.7	**40.6**	40.7	41.2	41.3
WER(%) (E3 SLM)	41.8	40.7	40.7	40.7	40.7	41.2	41.3

Conclusions

- Approach to language modeling that takes into account the hierarchical structure in natural language
- Algorithm to re-estimate the model parameters to decrease the perplexity of the model
- (Small) improvement in perplexity over trigram

Exercise

Consider the following parsed sentence from a treebank:

```
(S (NP (NNP pierre) (NNP vinken) )
   (VP (MD will)
       (VP (VB join)
           (NP (DT the) (NN board) )
           (PP (IN as)
               (NP (DT a) (JJ nonexecutive) (NN director) ) ) ) ) )
```

a) Construct the parse tree of the sentence and percolate the headwords using the rules given in Table 2.1 (handouts) / Table I (paper). Which word will be percolated to the root node `S`?

b) Give a partial parse (cf. Figure 2.5 in handouts / Figures 1. and 2. in the paper) of the beginning of the sentence up to the word `board`.

c) Suppose we are predicting the next word after `board` using SLM. What words does the model use for the prediction?