Exploiting Syntactic Structure for Language Modeling

Ciprian Chelba, Frederick Jelinek

- Basic Language Modeling:

  A Structured Language Model:
  - Language Model Requirements
  - Word and Structure Generation
  - Research Issues:
    - Model Component Parameterization
    - Pruning Strategy
    - Word Level Probability Assignment
    - Model Statistics Reestimation
  - Model Performance
• give people an outline so that they know what’s going on
Basic Language Modeling

Estimate the source probability

\[ P(W), \quad W = w_1, w_2, \ldots, w_n \]

from a training corpus — millions of words of text chosen for its similarity to the sentences expected at run-time.

Parametric conditional models

\[ P_\theta(w_i | w_1 \ldots w_{i-1}), \theta \in \Theta, w_i \in \mathcal{V} \]

Perplexity (PPL)

\[
PPL(M) = \exp \left( -\frac{1}{N} \sum_{i=1}^{N} \ln [ P_M(w_i | w_1 \ldots w_{i-1}) ] \right)
\]

✔ different than maximum likelihood estimation: the test data is not seen during the model estimation process;

✔ good models are smooth:

\[ P_M(w_i | w_1 \ldots w_{i-1}) > \epsilon \]
- Source modeling; classical problem in information theory
- give interpretation for perplexity as expected average length of list of equi-probable words; Shannon code-length;
Exploiting Syntactic Structure for Language Modeling

- Generalize trigram modeling (local) by taking advantage of sentence structure (influence by more distant past)
- Use exposed heads $h$ (words $w$ and their corresponding non-terminal tags $l$) for prediction:

$$P(w_i|T_i) = P(w_i|h_{-2}(T_i), h_{-1}(T_i))$$

$T_i$ is the partial hidden structure, with head assignment, provided to $W_i$
• point out originality of approach;
• explain clearly what headwords are;
• difference between trigram/slm: surface/deep modeling of the source; give example with removed constituent again; show that they make intuitively better predictors for the following word;
• hidden nature of the parses; cannot decide on a single best parse for a word prefix, not even at the end of sentence;
• need to weight them according to how likely they are - probabilistic model;
Language Model Requirements

- Model must operate left-to-right: \( P(w_i/w_1 \ldots w_{i-1}) \)
- In hypothesizing hidden structure, the model can use only word-prefix \( W_i \), i.e., not the complete sentence \( w_0, \ldots, w_i, \ldots, w_{n+1} \) as all conventional parsers do!
- Model complexity must be limited; even trigram model faces critical data sparseness problems
- Model will assign joint probability to sequences of words and hidden parse structure:

\[
P(T_i, W_i)
\]
...; null; predict cents;
the_DT contract_NN ended_VBD with_IN a_DT loss_NN of_IN 7_CD cents_NNS
The contract ended with a loss of the contract 7 cents; null; predict cents; POStag cents; adjoin-right-NP;
ended with a loss of the contract 7 cents

...; null; predict cents; POStag cents; adjoin-right-NP; adjoin-left-PP;
ended_VBD with_IN a_DT loss_NN of_IN the_DT contract_NN contract_NP 7_CD cents_NNS of_PP with_PP loss_NP loss_NP of_IN cents_NP cents_NP null; predict cents; POS_tag cents; adjoin-right-NP; adjoin-left-PP; ...; adjoin-left-VP'; null; ...;
• just one of the possible continuations for one of the possible parses of the prefix;
• prepare next slide using FSM; explain that it is merely an encoding of the word prefix and the tree structure;
Word and Structure Generation

\[ P(T_{n+1}, W_{n+1}) = \prod_{i=1}^{n+1} P(w_i|h_{-2}, h_{-1}) P(g_i|w_i, h_{-1}.tag, h_{-2}.tag) P(T_i|w_i, g_i, T_{i-1}) \]

- The **predictor** generates the next word \( w_i \) with probability \( P(w_i = v|h_{-2}, h_{-1}) \)
- The **tagger** attaches tag \( g_i \) to the most recently generated word \( w_i \) with probability \( P(g_i|w_i, h_{-1}.tag, h_{-2}.tag) \)
- The **parser** builds the partial parse \( T_i \) from \( T_{i-1}, w_i, \) and \( g_i \) in a series of *moves* ending with \textbf{null}, where a parser move \( a \) is made with probability \( P(a|h_{-2}, h_{-1}); \)
  \[ a \in \{ \text{adjoin-left, NTtag}, \text{adjoin-right, NTtag}, \text{null} \} \]
• we have described an encoding of a word sequence with a parse tree;
• to get a probabilistic model assign a probability to each elementary action in the encoding
Research Issues

- Model component parameterization — equivalence classifications for model components:
  \[ P(w_i = v|h_{-2}, h_{-1}), P(g_i|w_i, h_{-1}.tag, h_{-2}.tag), P(a|h_{-2}, h_{-1}) \]

- Huge number of hidden parses — need to prune it by discarding the unlikely ones

- Word level probability assignment — calculate \( P(w_i/w_1 \ldots w_{i-1}) \)

- Model statistics estimation — unsupervised algorithm for maximizing \( P(W) \) (minimizing perplexity)
everything's on the slide
**Pruning Strategy**

Number of parses $T_k$ for a given word prefix $W_k$ is $|\{T_k\}| \sim O(2^k)$;

Prune most parses without discarding the most likely ones for a given sentence

**Synchronous Multi-Stack Pruning Algorithm**

- the hypotheses are ranked according to $\ln(P(W_k, T_k))$
- each stack contains partial parses constructed by *the same number of parser operations*

The width of the pruning is controlled by:

- maximum number of stack entries
- log-probability threshold
Pruning Strategy

(k) 0 parser op k predict. 0 parser op k predict. 0 parser op k predict.

(k') 0 parser op k+1 predict. 0 parser op k+1 predict. 0 parser op k+1 predict.

(k+1) 0 parser op k+1 predict. 0 parser op k+1 predict. 0 parser op k+1 predict.

... ... ...

p parser op k predict. p parser op k predict. p parser op k predict.

p+1 parser k predict. p+1 parser k predict. p+1 parser k predict.

... ... ...

P_k parser k predict. P_k parser k predict. P_k parser k predict.

P_k+1 parser k+1 predict. P_k+1 parser k+1 predict. P_k+1 parser k+1 predict.

... ... ...

null parser transitions parser adjoin/unary transitions

word predictor and tagger
• we want to find the most probable set of parses that are extensions of the ones currently in the stacks
• there is an upper bound on the number of stacks at a given input position
• hypotheses in stack 0 differ according to their POS sequences
Word Level Probability Assignment

The probability assignment for the word at position \( k + 1 \) in the input sentence must be made using:

\[
P(w_{k+1}/W_k) = \sum_{T_k \in S_k} P(w_{k+1}/W_kT_k) \cdot \rho(W_k, T_k)
\]

- \( S_k \) is the set of all parses present in the stacks at the current stage \( k \)
- interpolation weights \( \rho(W_k, T_k) \) must satisfy:

\[
\sum_{T_k \in S_k} \rho(W_k, T_k) = 1
\]

in order to ensure a proper probability over strings \( W^* \):

\[
\rho(W_k, T_k) = P(W_kT_k) / \sum_{T_k \in S_k} P(W_kT_k)
\]
• point out consistency of estimate: when summing over all parses we get the actual probability value according to our model.
Model Parameter Reestimation

Need to re-estimate model component probabilities such that we decrease the model perplexity.

\[ P(w_i = v| h_{-2}, h_{-1}), P(g_i| w_i, h_{-1}.tag, h_{-2}.tag), P(a|h_{-2}, h_{-1}) \]

Modified Expectation-Maximization (EM) algorithm:

- We retain the \( N \) “best” parses \( \{T^1, \ldots, T^N\} \) for the complete sentence \( W \)
- The hidden events in the EM algorithm are restricted to those occurring in the \( N \) “best” parses
- We seed re-estimation process with statistics gathered from manually parsed sentences
- point out goal of re-estimation
- warn about need to know the E-M algorithm;
- explain what a treebank is and why/how we can initialize from treebank
Language Model Performance — Perplexity

- Training set: UPenn Treebank text; 930Kwds; manually parsed;
- Test set: UPenn Treebank text; 82Kwds;
- Vocabulary: 10K — out of vocabulary words are mapped to <unk>
- incorporate trigram in word PREDICTOR:

\[
P(w_i|W_i) = (1 - \lambda) \cdot P(w_i|h_{-2}, h_{-1}) + \lambda \cdot P(w_i|w_{i-1}, w_{i-2}), \lambda = 0.36
\]

<table>
<thead>
<tr>
<th>Language Model</th>
<th>L2R Perplexity</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DEV set</td>
</tr>
<tr>
<td></td>
<td>no int</td>
</tr>
<tr>
<td>Trigram</td>
<td></td>
</tr>
<tr>
<td>(P(w_i</td>
<td>w_{i-2}, w_{i-1}))</td>
</tr>
<tr>
<td>Seeded with Treebank</td>
<td></td>
</tr>
<tr>
<td>(P_0(w_i</td>
<td>h_{i-2}, h_{i-1}))</td>
</tr>
<tr>
<td>Reestimated</td>
<td></td>
</tr>
<tr>
<td>(P(w_i</td>
<td>h_{i-2}, h_{i-1}))</td>
</tr>
</tbody>
</table>

The Johns Hopkins University
• first model that reports a reduction over trigram model by using syntactic structure
• make point about data over-fitting in the trigram case — caused by data sparseness and poor source modeling (surface model);
Conclusion

✔ original approach to language modeling that takes into account the hierarchical structure in natural language

✔ devised an algorithm to reestimate the model parameters such that the perplexity of the model is decreased

✔ showed improvement in perplexity over current language modeling techniques

Future Work

✘ rescoring of word lattices output by a speech recognizer
• BOW!
Exploiting Syntactic Structure for Language Modeling

Ciprian Chelba, Frederick Jelinek

Acknowledgments:

- this research was funded by the NSF grant IRI-19618874 (STIMULATE);
- thanks to Eric Brill, William Byrne, Sanjeev Khudanpur, Harry Printz, Eric Ristad, Andreas Stolcke and David Yarowsky for useful comments, discussions on the model and programming support