Augmenting recurrent neural network language models with subword information

Lyan Verwimp, Joris Pelemans, Hugo Van hamme, Patrick Wambacq

lyan.verwimp@esat.kuleuven.be
Introduction
Language models

- **Recurrent neural network** LMs (RNNLMs) (Mikolov et al. 2010)

![Diagram of a recurrent neural network language model](image)

Input (t) → Hidden (t) → Output (t)

- Input (t) represents a sequence of characters, e.g., "man".
- Hidden (t) is the internal state of the network at time step t.
- Output (t) is the predicted output word.

```
man → 0 1 0 ...
\[\vdots\]
\[\vdots\]
0 0 0
```

```
Hidden (t-1) → Hidden (t)
```

```
Output (t) → \[\vdots\]
```

KU LEUVEN
Language models

- *N*-gram LMs: $P(w_n|w_1, \ldots, w_{n-1}) \approx P(w_n|w_{n-N+1}, \ldots, w_{n-1})$
  - 😊 Efficient
  - 😞 Fast to train and evaluate
  - ☹️ Data sparsity
  - 😤 No long-span dependencies

- RNNLMs
  - 😊 Word represented as continuous vector: more generalization
  - 😊 Recurrence: long-span dependencies
  - ☹️ Computationally more complex
  - 😤 Not in first pass of speech recognition
Motivation
Character-Word RNNLMs

Input (t) -> Hidden (t) -> Output (t)

Characters -> Hidden (t-1)
Character-Word RNNLMs

Characters → .... → e.g. 

\[ \begin{align*} 
    & m \\
    & 0 \ldots 1 \ldots 0 \\
    & a \\
    & 1 \ldots 0 \ldots 0 \ldots 0 \\
    & n \\
    & 0 \ldots 0 \ldots 1 \ldots \end{align*} \]
Motivation

• Input of RNNLM = one-hot vector of word
  ➢ Generalize over similar contexts
    e.g. man – vrouw

• Our approach: add information about formal/morphological structure of the word
  ➢ Also generalize over words with similar formal/morphological structure
    e.g. man – huisman
  ➢ OOV-words: subwords can still give meaningful information
Subword information

- **Characters**
  - Easy to extract
  - Always correct
  - ‘vocabulary’ (alphabet) is small: few extra parameters
  - No meaningful units
    - Normal order + inverted order
    - How many?

- **Later stage: probably morphemes**
  - Meaningful units
  - More difficult to extract
  - More prone to errors
Our RNNLMs:

- **Hidden layer:**
  - 300 units
  - Sigmoid activation function:
    \[ f(z) = \frac{1}{1+e^{-z}} \]

- **Output layer:**
  - 100 classes (based on frequency)
  - Softmax activation function:
    \[ g(z_m) = \frac{e^{z_m}}{\sum_k e^{z_k}} \]

- **Backpropagation through time:** 5 time steps
Experiments
Perplexity experiments: Setup

• Corpus: CGN components g, h, n, o
  o Training: 6,2M word tokens (vocabulary: 125k)
  o Validation: 1,3M word tokens
  o Test: 860k word tokens

• RNNLMs: RNNLM toolkit (Mikolov et al. 2011): extension by Shi et al. (2015) to incorporate features
Perplexity experiments

![Perplexity experiments graph showing the relationship between perplexity and the number of character vectors, with lines representing different orders: baseline, normal order, reversed order, and both. The graph shows a steady increase in perplexity as the number of character vectors increases.]
Perplexity experiments

<table>
<thead>
<tr>
<th>Normal order</th>
<th>Reversed order</th>
<th>RNNLM - both</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>136</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>134</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>135</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>133</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>136</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>135</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>135</td>
</tr>
</tbody>
</table>

= baseline
Speech recognition experiments: Setup

- Multi-pass speech recognition:
  - First pass: lattices $\rightarrow$ $N$-best lists
    - Language model trained on 1,4B word tokens (vocabulary = 400k)
    - SPRAAK (Demuynck et al. 2008, 2009)
  - Second pass: rescore $N$-best lists
    - Shi’s extension of the RNNLM toolkit: rescoring
    - SPRAAK: alignment

- Test set: 76k word tokens
# Speech recognition experiments

<table>
<thead>
<tr>
<th>Model</th>
<th># of character vectors</th>
<th>WER (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>/</td>
<td>48.8</td>
</tr>
<tr>
<td>Normal order</td>
<td>3</td>
<td>58</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>59</td>
</tr>
<tr>
<td>Reversed order</td>
<td>3</td>
<td>57.5</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>59.1</td>
</tr>
<tr>
<td>Both</td>
<td>3 – 2</td>
<td>55.1</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>59.7</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>59.8</td>
</tr>
</tbody>
</table>
Discussion and future work
Discussion

• Perplexity: some promising improvements
  o BUT only beginning and/or ending of word modeled
  e.g. barkruk and krukje

• Speech recognition: no improvements (yet)
  o But similar conclusions as for perplexity
  o Problems:
    • Capitals
    • Toolkit
    • Implementation: only one-hot vectors
**Future work**

- Other implementation
- Explore different ways of adding subword information
- LSTM language models
- Other subword information (morphemes/syllables)
- Larger training set
Thank you for your attention!
Questions?
References

Language models for speech recognition

• *hij leest een …*
  o *doek?*
  o *boek?*
  o *koek?*
  o *…?*
Language models for speech recognition

• *hij leest een* …
  
  o *doek?*  \[ P(\text{doek}|\text{hij leest een}) = 0.0002 \]
  
  o *boek?*  \[ P(\text{boek}|\text{hij leest een}) = 0.04 \]
  
  o *koek?*  \[ P(\text{koek}|\text{hij leest een}) = 0.0001 \]
  
  o *…?*  \[ P(\ldots|\text{hij leest een}) = \ldots \]

• *N-gram* LMs
  \[ P(w_1, \ldots, w_n) = P(w_1|w_2, \ldots, w_n) = P(w_1|w_{n-N+1}) \]
**RNN vs LSTM**

*Figure 1. Detailed schematic of the Simple Recurrent Network (SRN) unit (left) and a Long Short-Term Memory block (right) as used in the hidden layers of a recurrent neural network.*
Content

• Introduction
  o Language models (LMs)
  o Recurrent neural network LMs
• Motivation
• Experiments
  o Setup
  o Results
• Discussion and future work
• References