Subword-based Compact Reconstruction of Word Embeddings

Shota Sasaki, Jun Suzuki, Kentaro Inui

RIKEN AIP, Tohoku University
Proposal: novel word embeddings

**OPEN-vocabulary**

- Previous 2M word
- Ours: Unlimited!

**COMPACT model size**

- Previous 2GB
- Ours: 200MB!

**Performance:**

- **Better score**
  - WordSim (OOV)

- **1/4 model size**
  - Comparable score
  - Model Compression Test

- **1/10 model size**
  - Comparable score
  - Downstream Tasks
• Quick Overview

• Background
  • Word embeddings
  • Related work
  • Purpose

• Proposed Method
  • Key technique
    • Subword-to-memory mapping function
    • Subword mixing function

• Experiments
  • Word similarity with OOV words
  • Model compression test
  • Downstream tasks (NER, TE)
Background: Pretrained Word Embeddings

✅ Highly beneficial, fundamental language resources
- e.g., GloVe.840B embeddings [Pennington, 2014]
  - Training data: Common Crawl Corpus (840B tokens)
  - Available online

❌ Inapplicability to out-of-vocabulary (OOV) words
- Infrequent words (often cut off due to memory requirements)
- Novel words
Background: Pretrained Word Embeddings

✓ Highly beneficial, fundamental language resources
  • e.g., GloVe.840B embeddings [Pennington, 2014]
  • Training data: Common Crawl Corpus (840B tokens)
  • Available online

✗ Inapplicability to out-of-vocabulary (OOV) words
  • Novel words
  • Infrequent words (often cut off due to memory requirements)
Related Work: Bag of Subwords (BoS) [Zhao et al., EMNLP-2018]

• Similar motivation
  Reconstruct pretrained word embeddings
to support out-of-vocabulary (OOV) words

• Basic Idea
  Compute embeddings of OOV words by summing up subword embeddings obtained through the reconstruction
The Problem with Subwords: There are too many

Naïve approach significantly increases memory requirements

<table>
<thead>
<tr>
<th>Setting</th>
<th># of vectors (aka. # of vocab.)</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-trained word embeddings (fastText.600B)</td>
<td>2.0 M</td>
<td>2.2 GB</td>
</tr>
<tr>
<td>char N-gram (N=1, 2, ..., 6) subword embeddings</td>
<td>6.3 M</td>
<td>7.2 GB</td>
</tr>
</tbody>
</table>

Mem. (GB) = # of vectors × # of dimensions × 4bytes (float) / 1024^3
Aim to develop a method that simultaneously satisfies
① less memory requirement  ② applicability of OOV
Outline

• Quick Overview

• Background
  • Word embeddings
  • Related work
  • Purpose

• Proposed Method
  • Key technique
    • Subword-to-memory mapping function
    • Subword mixing function

• Experiments
  • Word similarity with OOV words
  • Model compression test
  • Downstream tasks (NER, TE)
Outline

• Quick Overview

• Background
  • Word embeddings
  • Related work
  • Purpose

• Proposed Method
  • Key technique
    • Subword-to-memory mapping function
    • Subword mixing function

• Experiments
  • Word similarity with OOV words
  • Model compression test
  • Downstream tasks (NER, TE)
Key Ideas

- Subword-to-memory mapping function
  1. Discarding infrequent subwords
  2. Memory sharing
  3. Combination of 1. and 2.

- Subword mixing function
  • Self-attention mechanism
Key Ideas

- Subword-to-memory mapping function
  1. Discarding infrequent subwords
  2. Memory sharing
  3. Combination of 1. and 2.

- Subword mixing function
  - Self-attention mechanism
1. Discarding infrequent subwords
   - Use top-$F$ frequent subwords instead of all possible subwords
   - Model size $= F \times \# \text{ of dimensions}$
Subword-to-Memory Mapping

1. Discarding infrequent subwords
   - Use top-\( F \) frequent subwords instead of all possible subwords
   - Model size = \( F \times \# \) of dimensions
1. Discarding infrequent subwords
   - Use top-$F$ frequent subwords instead of all possible subwords
   - Model size $= F \times \# \text{ of dimensions}$

2. Memory sharing [Bojanowski, 2017]
   - Randomly share the same vectors between several subwords
   - Model size $= H \times \# \text{ of dimensions}$
1. Discarding infrequent subwords
   - Use top-\(F\) frequent subwords instead of all possible subwords
   - Model size = \(F \times \# \text{ of dimensions}\)

2. Memory sharing [Bojanowski, 2017]
   - Randomly share the same vectors between several subwords
   - Model size = \(H \times \# \text{ of dimensions}\)
3. Combination
   I. Reduce subword vocabulary to top-$F$ frequent subwords
   II. Apply memory sharing method
Subword-to-Memory Mapping

3. Combination
   I. Reduce subword vocabulary to top-$F$ frequent subwords
   II. Apply memory sharing method
3. Combination
   I. Reduce subword vocabulary to top-$F$ frequent subwords
   II. Apply memory sharing method
Key Ideas

- Subword-to-memory mapping function
  1. Discarding infrequent subwords
  2. Memory sharing
  3. Combination of 1. and 2.

- Subword mixing function
  - Self-attention mechanism
Key Ideas

- Subword-to-memory mapping function
  1. Discarding infrequent subwords
  2. Memory sharing
  3. Combination of 1. and 2.

- Subword mixing function
  - Self-attention mechanism
Subword Mixing Function

Subwords:
- h
- i
- low
- er
- ighe
- ower...

Memory:

Mixing function

Result: ‘higher’
Subword Mixing Function

The diagram illustrates the process of subword mixing in word embeddings. It shows a series of subwords connected to a memory module, which then feeds into a summation function. The output of the summation is labeled as ‘higher’.
Subword Mixing Function

- Risky in a memory sharing setting where subwords randomly share the same vector
Subword Mixing Function

Subwords

- h
- i
- low
- er
- ighe
- ower
- ...

Memory

Key-value-query

‘higher’
Key-value-query self-attention operation

- incorporate a “context-dependent” weighting factor \( a_{s,w} \)
- “context” = all the subwords obtained from word \( w \)

\[
\tau_{\text{sum}}(V, w) = \sum_{s \in \phi(w)} v_s.
\]

\[
\tau_{\text{kvq}}(V, w) = \sum_{s \in \phi(w)} a_{s,w} v_s.
\]
Modification of Mixing Function - Key-value-query Operation

- Query vectors: \( h \) + \( i \) + ... + \( er \)
- Key vectors: \( \alpha_h \) \( \times \) \( \alpha_i \) \( \times \) \( \alpha_{er} \)
- Value vectors: + + + ...

\( \odot \) dot product  \( \otimes \) scalar multiplication

\( 'higher' \)
Modification of Mixing Function
- Key-value-query Operation

'higher'

Query vectors

Key vectors

Value vectors
Modification of Mixing Function
- Key-value-query Operation

Query vectors: $h + i + \ldots + er$

Key vectors: $\alpha_h \times \alpha_i \times \alpha_{er}$

Value vectors: $\odot$ dot product

Scalar multiplication: $\otimes$

'higher'
Modification of Mixing Function
- Key-value-query Operation

Query vectors

Key vectors

Value vectors

更高的

\[ H = \alpha_h + \alpha_i + \cdots + \alpha_{er} \]

\( \odot \) dot product

\( \otimes \) scalar multiplication
Modification of Mixing Function
- Key-value-query Operation

Query vectors + + ... +

Key vectors

Value vectors

‘higher’

\( h \quad i \quad \ldots \quad er \)

\( H \)

\( \alpha_h \quad \alpha_i \quad \ldots \quad \alpha_{er} \)

\( \bigodot \) dot product

\( \bigotimes \) scalar multiplication

2019/06/05

Subword-based Compact Reconstruction of Word Embeddings
Advantages

• Highly expressive
  • allows assigning a lower weight to subword vector sharing its memory with completely unrelated subword

• No need of extra transformation matrix
  • Model size = $H \times \# \text{ of dimensions}$
Key Ideas

- Subword-to-memory mapping function
  1. Discarding infrequent subwords
  2. Memory sharing
  3. Combination of 1. and 2.

- Subword mixing function
  • Self-attention mechanism
Outline

• Quick Overview

• Background
  • Word embeddings
  • Related work
  • Purpose

• Proposed Method
  • Key technique
    • Subword-to-memory mapping function
    • Subword mixing function

• Experiments
  • Word similarity with OOV words
  • Model compression test
  • Downstream tasks (NER, TE)
### Evaluation of OOV Word Embeddings

- **Word Similarity (Rare Word dataset)**
  - Followed experimental settings used in [Zhao, EMNLP-2018]
  - 2000 word pairs, OOV rate: 11%

<table>
<thead>
<tr>
<th>Method</th>
<th>Spearman’s $\rho$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Baseline</strong></td>
<td></td>
</tr>
<tr>
<td>Random</td>
<td>0.452</td>
</tr>
<tr>
<td>BoS [Zhao, EMNLP-2018]</td>
<td>0.46*</td>
</tr>
<tr>
<td><strong>Proposed</strong></td>
<td></td>
</tr>
<tr>
<td>SUM-topF</td>
<td>0.513</td>
</tr>
<tr>
<td>SUM-share</td>
<td>0.485</td>
</tr>
<tr>
<td>KVQ-share</td>
<td>0.509</td>
</tr>
<tr>
<td>SUM-comb</td>
<td>0.488</td>
</tr>
<tr>
<td><strong>KVQ-comb</strong></td>
<td><strong>0.522</strong></td>
</tr>
</tbody>
</table>

- ✓ Our methods outperformed previous method
Evaluation of Model Compression

• Evaluation tasks
  • Word similarity task (9 datasets)

• Pre-trained Embeddings
  • fastText embeddings trained on Common Crawl corpus
    • 2M words, 300 dimensions

• Note: discarded pairs containing at least one OOV word
Results on Word Similarity Task

Spearman Correlation

Number of subword embeddings (× 1000)

Pretrained Embeddings (# of vectors = 2000k)

SUM-topF

200 300 400 500

0.50 0.55 0.60 0.65 0.70 0.75

2019/06/05

Subword-based Compact Reconstruction of Word Embeddings
Results on Word Similarity Task

Pretrained Embeddings (# of vectors = 2000k)

- SUM-topF
- SUM-share

**Number of subword embeddings (×1000)**

**Spearman Correlation**

- 0.50
- 0.55
- 0.60
- 0.65
- 0.70
- 0.75
Results on Word Similarity Task

Pretrained Embeddings (# of vectors = 2000k)

Spearman Correlation

Number of subword embeddings (×1000)

- KVQ-share
- SUM-topF
- SUM-share

2019/06/05
Subword-based Compact Reconstruction of Word Embeddings
Results on Word Similarity Task

Spearman Correlation

Pretrained Embeddings (# of vectors = 2000k)

- KVQ-comb
- KVQ-share
- SUM-topF
- SUM-share

Number of subword embeddings (× 1000)

- KVQ-comb achieved comparable performance with less memory requirements
Evaluation on Downstream Tasks

- Used AllenNLP implementation, default settings

### Textual Entailment (SNLI)

<table>
<thead>
<tr>
<th></th>
<th>Size (GB)</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>fastText word emb.</td>
<td>2.23GB</td>
<td>87.8</td>
</tr>
<tr>
<td>KVQ-comb (H=0.5M)</td>
<td>0.59GB</td>
<td>88.0</td>
</tr>
<tr>
<td>KVQ-comb (H=0.2M)</td>
<td>0.23GB</td>
<td>87.6</td>
</tr>
</tbody>
</table>

### Named Entity Recognition (CoNLL-2003)

<table>
<thead>
<tr>
<th></th>
<th>Size (GB)</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>fastText word emb.</td>
<td>2.23GB</td>
<td>90.3</td>
</tr>
<tr>
<td>KVQ-comb (H=0.5M)</td>
<td>0.59GB</td>
<td>90.4</td>
</tr>
<tr>
<td>KVQ-comb (H=0.2M)</td>
<td>0.23GB</td>
<td>89.3</td>
</tr>
</tbody>
</table>

✓ KVQ-comb achieved comparable performance with less memory requirements
Conclusion

Proposal: novel word embeddings

OPEN-vocabulary

- Previous 2M word
- ours Unlimited!!

COMPACT model size

- Previous 2GB
- ours 200MB!!

Performance:

- ✔ Better score
- ✔ 1/4 model size
- ✔ 1/10 model size
- ✔ Comparable score
- ✔ Comparable score

WordSim (OOV)
Model Compression Test
Downstream Tasks