Natural Language Processing

Subword models

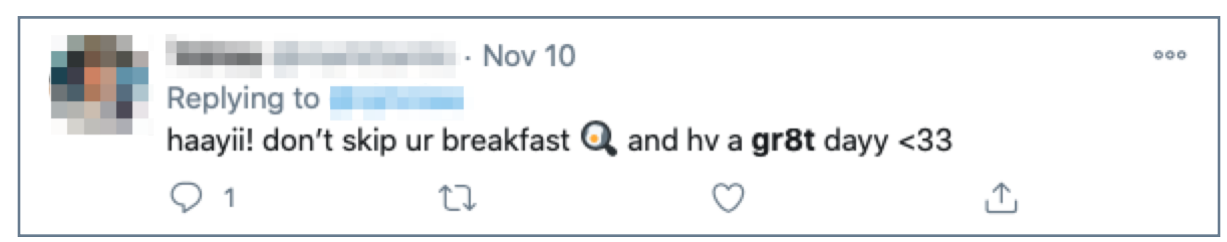
Marco Kuhlmann
Department of Computer and Information Science

LINKÖPING UNIVERSITY

Subword models

- Word embeddings as we have covered them so far assume atomic words and a fixed vocabulary.
- In practical applications, we will often encounter words that we do not have an embedding for.

  Remember Heaps' law!



- One way to deal with this problem is to use models that work at the subword level, such as character-based models.

Rationale for subword models

- Working with subword units makes sense from a linguistic point of view, as subword units resemble morphemes.

  Morpheme+s are the small+est mean+ing+ful unit+s of language.

- Features at the subword level have been shown to be very predictive in non-neural models for e.g. part-of-speech tagging.

  Does the word end in *-tion* or *-ism*? Then chances are, it's a noun!

Different types of subword models

- **Type 1:** Use the same types of architectures that we find in word-based models, but apply them to subword units.
- **Type 2:** Augment the architectures of word-based models with submodels that compose word representations from characters.
- **Type 3:** Give up on word-based architectures altogether and process language as a connected sequence of characters.

WordPiece tokenisation in BERT


The Muppet Wiki

Raw text

The history of morphological analysis dates back to the ancient Indian linguist Pāṇini, who formulated the 3,959 rules of Sanskrit morphology in the text Aṣṭādhyāyī by using a constituency grammar.

WordPiece tokenisation

The history of m ##or ##phological analysis dates back to the ancient Indian linguist P ##ā ##ṇ ##ini , who formulated the 3 , 95 ##9 rules of Sanskrit morphology in the text A ##ṣ ##ṭ ##ā ##dh ##y ##ā ##y ##ī by using a constituency grammar .

To obtain a word vector, take the average of the 9 word piece vectors.

Byte Pair Encoding algorithm

- Initialise the word unit vocabulary with all characters.

  plus a special end-of-word marker, here denoted by $

- Generate a new word unit by combining two units from the current vocabulary, increasing vocabulary size by one. Choose the new unit as the most frequent pair of adjacent units.

  WordPiece: maximise likelihood under a language model

- Repeat the previous step as long as the vocabulary size does not exceed a maximal size.

Sennrich et al. (2016); Schuster and Nakajima (2012)

Byte Pair Encoding: Example

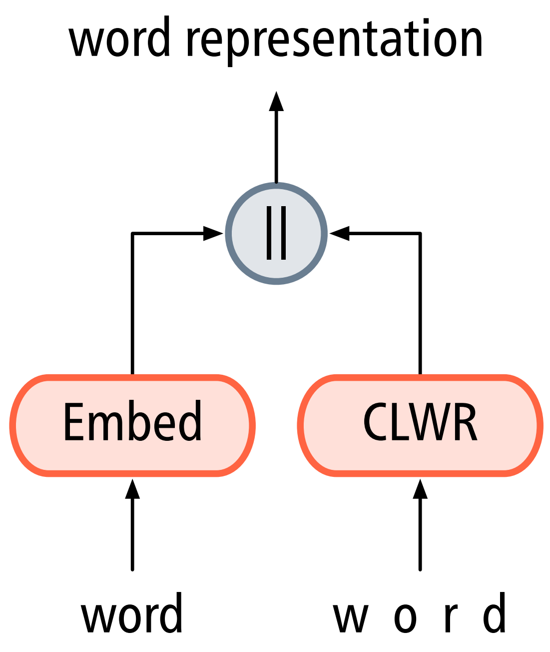
number of occurrences in data

Step	Merged pair	Words	Vocabulary size
0	–	low$/5 lower$/2 new**est**$/6 wid**est**$/3	11
1	es/9	low$ lower$ new**[es]t**$ wid**[es]t**$	12
2	[es]t/9	low$ lower$ new**[est]$** wid**[est]$**	13
3	[est]$/9	**lo**w$ **lo**wer$ new[est$] wid[est$]	14
4	lo/7	**[lo]w**$ **[lo]w**er$ new[est$] wid[est$]	15
5	[lo]w/7	[low]$ [low]er$ **ne**w[est$] wid[est$]	16

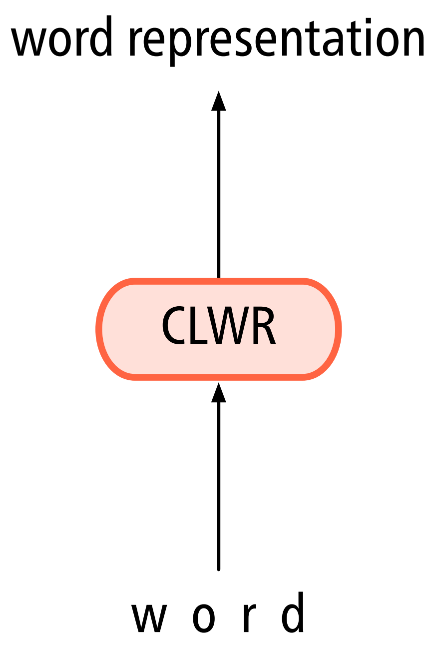
Example from Sennrich et al. (2016)

Composing word representations from characters

Character-level word representations are typically built using convolutional neural networks or recurrent neural networks.



combined (augmented) model



purely character-based model

Composing word representations using CNNs

<pad>	0.00 0.50	0.00 0.10	0.00 0.10
d	0.08 1.00	0.95 0.20	0.85 0.20
o	0.98 0.50	0.78 0.10	0.02 0.10
c	0.32	0.13	0.82
t	0.64	0.28	0.92
o	0.05	0.25	0.77
r	0.88	0.59	0.66
<pad>	0.00	0.00	0.00

1.010		
1.615		
1.520		
1.262		
1.259		
1.257		

Composing word representations using CNNs

<pad>	0.00 0.10	0.00 0.50	0.00 0.10
d	0.08 0.20	0.95 1.00	0.85 0.20
o	0.98 0.10	0.78 0.50	0.02 0.10
c	0.32	0.13	0.82
t	0.64	0.28	0.92
o	0.05	0.25	0.77
r	0.88	0.59	0.66
<pad>	0.00	0.00	0.00

1.010	1.626	
1.615	1.727	
1.520	1.144	
1.262	0.978	
1.259	1.159	
1.257	1.050	

Composing word representations using CNNs

<pad>	0.00 0.10	0.00 0.10	0.00 0.50
d	0.08 0.20	0.95 0.20	0.85 1.00
o	0.98 0.10	0.78 0.10	0.02 0.50
c	0.32	0.13	0.82
t	0.64	0.28	0.92
o	0.05	0.25	0.77
r	0.88	0.59	0.66
<pad>	0.00	0.00	0.00

1.010	1.626	1.242
1.615	1.727	1.355
1.520	1.144	1.648
1.262	0.978	1.974
1.259	1.159	1.859
1.257	1.050	1.369

Composing word representations using CNNs

<pad>	0.00	0.00	0.00
d	0.08	0.95	0.85
o	0.98	0.78	0.02
c	0.32	0.13	0.82
t	0.64	0.28	0.92
o	0.05	0.25	0.77
r	0.88	0.59	0.66
<pad>	0.00	0.00	0.00

	1.010	1.626	1.242
	1.615	1.727	1.355
	1.520	1.144	1.648
	1.262	0.978	1.974
	1.259	1.159	1.859
	1.257	1.050	1.369
	max	*max*	*max*
character-level word representation	1.615	1.727	1.974

Training augmented models

- In augmented models, the character-level word representations let us deal with unknown words at test time.
- However, we need to actively encourage these models to learn these character-level representations at training time.
- In **word dropout**, we replace each word with a dummy ⟨UNK⟩ token with some dropout probability p, e.g.

$$p = \frac{\alpha}{\#(w) + \alpha} \qquad \text{where } \alpha \text{ is a small constant}$$

Kiperwasser and Goldberg (2016)