Distributed word representations: Static representations from contextual models

Christopher Potts

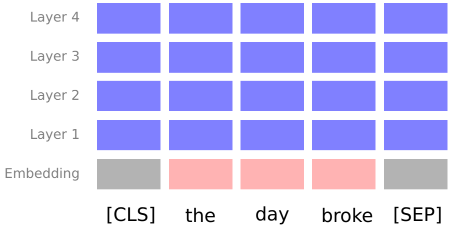
Stanford Linguistics

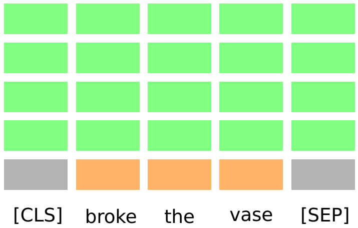
CS224u: Natural language understanding

Overview

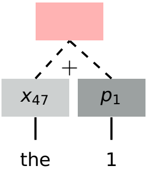
1. How can I use BERT (RoBERTa, XLNet, ELECTRA, . . .)?
2. Tension: we've been developing *static* representations, whereas BERT delivers *contextual* representations.
3. Are there good methods for deriving static representations from contextual ones?
4. Yes! Bommasani et al. (2020)!
5. This lecture:
   - Hands-on, high-level overview of these models. (We will analyze them in detail later in the quarter.)
   - Overview of the methods of Bommasani et al.

The structure of BERT





- The rectangles are vectors: the outputs of each layer of the network.
- Different sequences deliver different vectors for the same token, even in the embedding layer if the positions vary.



Tokenization

```
[1]: from transformers import BertTokenizer

[2]: tokenizer = BertTokenizer.from_pretrained('bert-base-cased')

[3]: tokenizer.tokenize("This isn't too surprising.")

[3]: ['This', 'isn', "'", 't', 'too', 'surprising', '.']

[4]: tokenizer.tokenize("Encode me!")

[4]: ['En', '##code', 'me', '!']

[5]: tokenizer.tokenize("Snuffleupagus?")

[5]: ['S', '##nu', '##ffle', '##up', '##agu', '##s', '?']

[6]: tokenizer.vocab_size

[6]: 28996
```

Basic Hugging Face interfaces

[1]:
```
import torch
from transformers import BertModel, BertTokenizer
```

[2]:
```
bert_weights_name = 'bert-base-cased'
```

[3]:
```
tokenizer = BertTokenizer.from_pretrained(bert_weights_name)
```

[4]:
```
model = BertModel.from_pretrained(bert_weights_name)
```

[5]:
```
ex = tokenizer.encode(
    "the day broke",
    add_special_tokens=True,
    return_tensors='pt')
ex
```

[5]:
```
tensor([[ 101, 1103, 1285, 2795,  102]])
```

[6]:
```
with torch.no_grad():
    reps = model(ex, output_hidden_states=True)
```

[7]:
```
# Embedding and then 12 layers:
len(reps.hidden_states)
```

[7]:
```
13
```

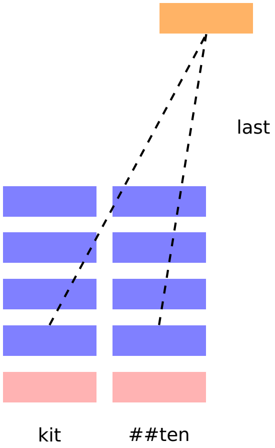
[8]:
```
# Embedding: batch of 1 example, 5 tokens, each represented
# by a vector of dimension 768:
reps.hidden_states[0].shape
```

[8]:
```
torch.Size([1, 5, 768])
```

[9]:
```
# Final output layer:
reps.hidden_states[-1].shape
```

[9]:
```
torch.Size([1, 5, 768])
```

The Decontextualized approach



Very simple! Potentially unnatural, though.

The Aggregated approach

Process *lots* of corpus examples containing the target word:

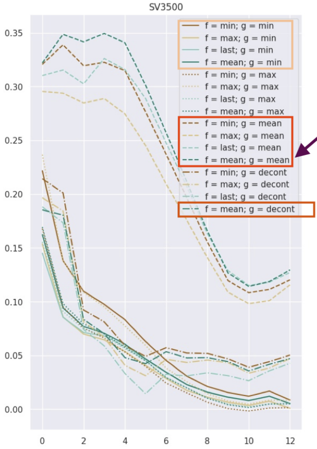
1. The kit ##ten yawned.
2. Where is my kit ##ten?
3. A kit ##ten is a young cat.
4. The puppy and the kit ##ten are playing.
5. ...

Pool sub-word tokens and pool the different contextuals reps.

A few results from Bommasani et al. (2020)

Lower layers best!

Spearman correlation



Consistently best

Layer

f: subword pooling
g: context pooling

References I

Rishi Bommasani, Kelly Davis, and Claire Cardie. 2020. Interpreting Pretrained Contextualized Representations via Reductions to Static Embeddings. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4758–4781, Online. Association for Computational Linguistics.