CS388: Natural Language Processing
Lecture 8: Pre-trained Encoders

Greg Durrett

TEXAS
The University of Texas at Austin

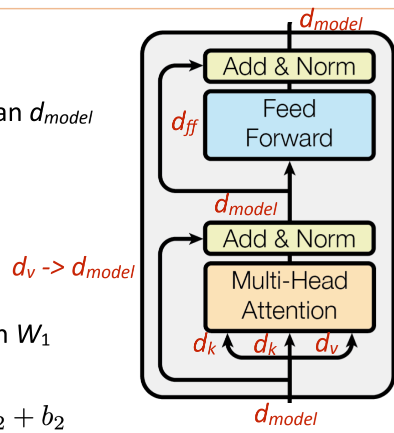
Announcements

- P2 due Tuesday
- P1 back tomorrow
- Final project released, proposals due Feb 23
  - Single or pairs, combining with other courses okay
  - Original research or reproduction
  - Topics, deliverables, etc. given in the spec
  - TACC allocation: send me usernames to be added. 4000 node-hours; usually 25% of the class uses this in a somewhat serious way, so ~10 groups

Recall: Transformers

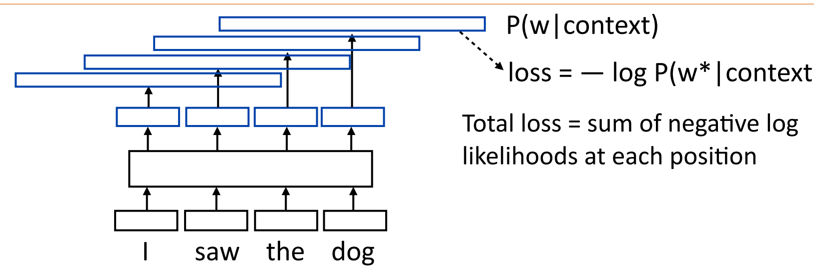
- Vectors: d_{model}
- Queries/keys: d_k, always smaller than d_{model}
- Values: separate dimension d_v, output is multiplied by W^O which is d_v x d_{model} so we can get back to d_{model} before the residual
- FFN can explode the dimension with W_1 and collapse it back with W_2

$$\mathrm{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$



Vaswani et al. (2017)

Recall: Training Transformer LMs



Total loss = sum of negative log likelihoods at each position

```
loss_fcn = nn.NLLLoss()
loss += loss_fcn(log_probs, ex.output_tensor)
```

[seq len, num output classes] [seq len]

- Batching is a little tricky with NLLLoss: need to collase [batch, seq len, num classes] to [batch * seq len, num classes]. You do not need to batch

Today

- ELMo
- BERT
- BERT results, BERT variants
- Subword tokenization

ELMo

What is pre-training?

- "Pre-train" a model on a large dataset for task X, then "fine-tune" it on a dataset for task Y
- Key idea: X is somewhat related to Y, so a model that can do X will have some good neural representations for Y as well
- ImageNet pre-training is huge in computer vision: learn generic visual features for recognizing objects
- GloVe can be seen as pre-training: learn vectors with the skip-gram objective on large data (task X), then fine-tune them as part of a neural network for sentiment/any other task (task Y)

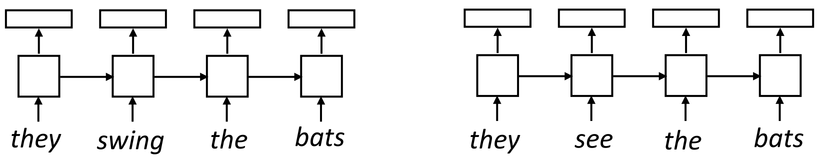
GloVe is insufficient

- GloVe uses a lot of data but in a weak way
- GloVe gives a single embedding for each word is wrong

  they swing the bats   *they see the bats*

  - Identifying discrete word senses is hard, doesn't scale. Hard to identify how many senses each word has
- How can we make our word embeddings more *context-dependent*? Use language model pretraining!

Context-dependent Embeddings

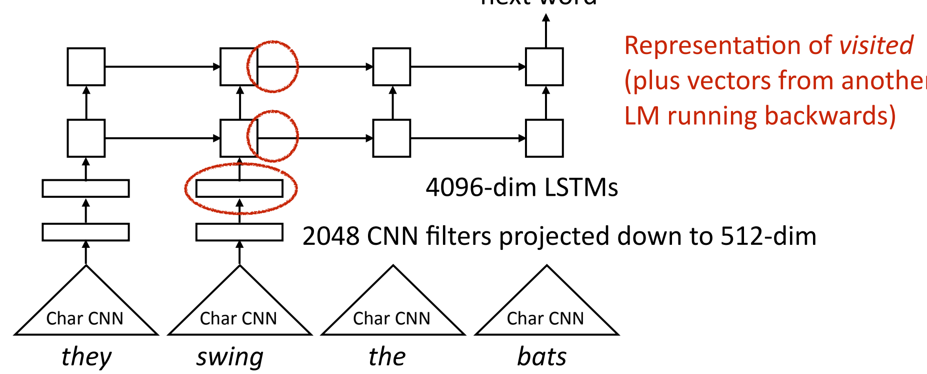


- Train a neural language model to predict the next word given previous words in the sentence, use the hidden states (output) at each step *as word embeddings*
- This is the key idea behind ELMo: language models can allow us to form useful word representations in the same way word2vec did

Peters et al. (2018)

ELMo

- CNN over each word => RNN



Peters et al. (2018)

ELMo

- Use the embeddings as a drop-in replacement for GloVe
- Huge gains across many high-profile tasks: NER, question answering, semantic role labeling (similar to parsing), etc.
- But what if the pre-training **isn't just for the embeddings?**

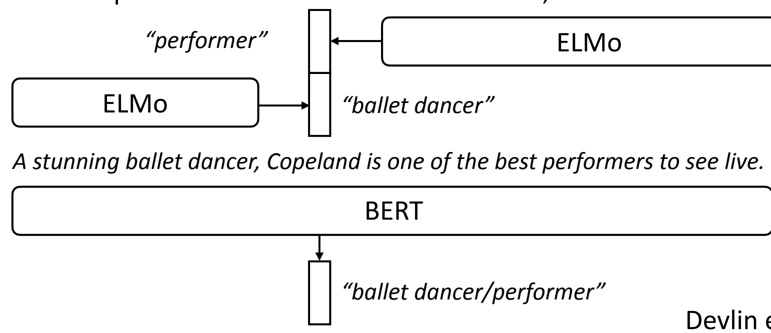
BERT

BERT

- AI2 made ELMo in spring 2018, GPT (transformer-based ELMo) was released in summer 2018, BERT came out October 2018
- Four major changes compared to ELMo:
  - Transformers instead of LSTMs
  - Bidirectional model with "Masked LM" objective instead of standard LM
  - Fine-tune instead of freeze at test time (**not just a source of word embeddings!**)
  - Operates over word pieces (byte pair encoding)

BERT

- ELMo is a unidirectional model (as is GPT): we can concatenate two unidirectional models, but is this the right thing to do?
- ELMo reprs look at each direction in isolation; BERT looks at them jointly

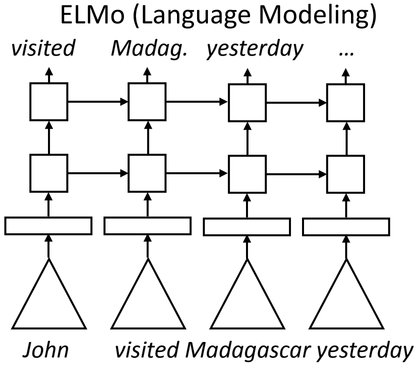


Devlin et al. (2019)

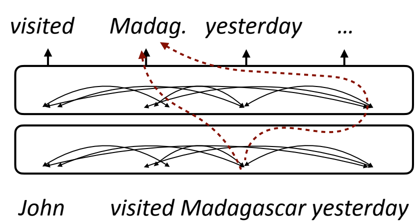
BERT

- How to learn a "deeply bidirectional" model? What happens if we just replace an LSTM with a transformer?

ELMo (Language Modeling)



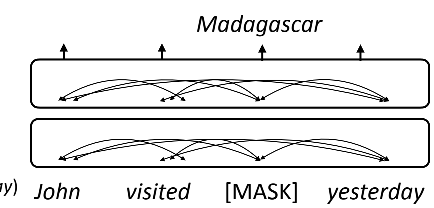
BERT



- You could do this with a "one-sided" transformer, but this "two-sided" model can cheat

Masked Language Modeling

- How to prevent cheating? Next word prediction fundamentally doesn't work for bidirectional models, instead do *masked language modeling*
- BERT formula: take a chunk of text, mask out 15% of the tokens, and try to predict them
- Optimize
  P(*Madagascar* | *John visited [MASK] yesterday*)



Devlin et al. (2019)

Next “Sentence” Prediction

- Input: [CLS] Text chunk 1 [SEP] Text chunk 2
- 50% of the time, take the true next chunk of text, 50% of the time take a random other chunk. Predict whether the next chunk is the “true” next
- **Why is this a good idea?**
- BERT objective: masked LM + next sentence prediction

NotNext *Madagascar* *enjoyed* *like*

Transformer

…

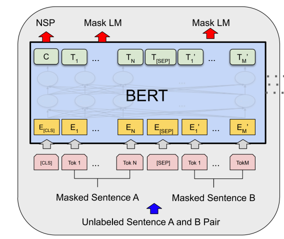
Transformer

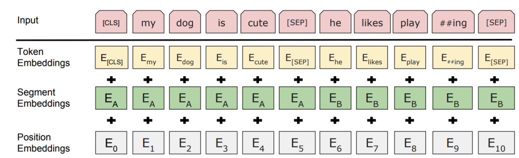
[CLS] *John visited* **[MASK]** *yesterday and really* **[MASK]** *it* [SEP] *I* **[MASK]** *Madonna.*

Devlin et al. (2019)

BERT Architecture

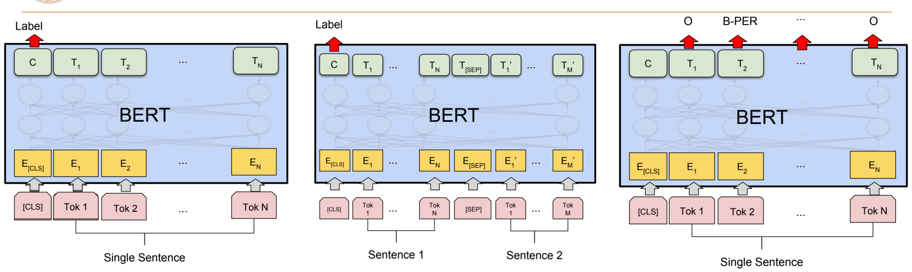
- BERT Base: 12 layers, 768-dim per wordpiece token, 12 heads. Total params = 110M
- BERT Large: 24 layers, 1024-dim per wordpiece token, 16 heads. Total params = 340M
- Positional embeddings and segment embeddings, 30k word pieces
- This is the model that gets **pre-trained** on a large corpus





Devlin et al. (2019)

What can BERT do?



(b) Single Sentence Classification Tasks: SST-2, CoLA

(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

- Artificial [CLS] token is used as the vector to do classification from
- Sentence pair tasks (entailment): feed both sentences into BERT
- BERT can also do tagging by predicting tags at each word piece

Devlin et al. (2019)

What can BERT do?

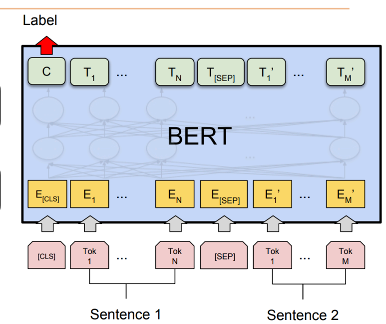
Entails (first sentence implies second is true)

Transformer

…

Transformer

[CLS] A boy plays in the snow [SEP] A boy is outside



(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

- How does BERT model sentence pairs?
- Transformers can capture interactions between the two sentences, even though the NSP objective doesn’t really cause this to happen

SQuAD

Q: What was Marie Curie the first female recipient of?

Passage: One of the most famous people born in Warsaw was Marie Skłodowska-Curie, who achieved international recognition for her research on radioactivity and was the first female recipient of the **Nobel Prize**. Famous musicians include Władysław Szpilman and Frédéric Chopin. Though Chopin was born in the village of Żelazowa Wola, about 60 km (37 mi) from Warsaw, he moved to the city with his family when he was seven months old. Casimir Pulaski, a Polish general and hero of the American Revolutionary War, was born here in 1745.

Answer = Nobel Prize

- Assume we know a passage that contains the answer. More recent work has shown how to retrieve these effectively (will discuss when we get to QA)

SQuAD

Q: What was Marie Curie the first female recipient of?

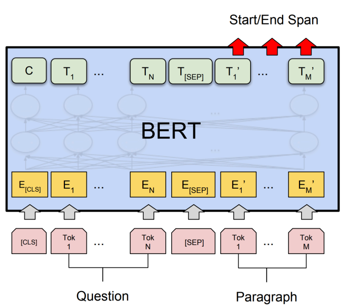
Passage: One of the most famous people born in Warsaw was Marie Skłodowska-Curie, who achieved international recognition for her research on radioactivity and was the first female recipient of the **Nobel Prize**. …

- Predict answer as a pair of (start, end) indices given question q and passage p; compute a score for each word and softmax those

	0.01	0.01	0.01	0.85	0.01
P(start | q, p) =	↑	↑	↑	↑	↑
	recipient	of	the	**Nobel**	**Prize** .

P(end | q, p) = same computation but different params

QA with BERT



What was Marie Curie the first female recipient of ? [SEP] One of the most famous people born in Warsaw was Marie …

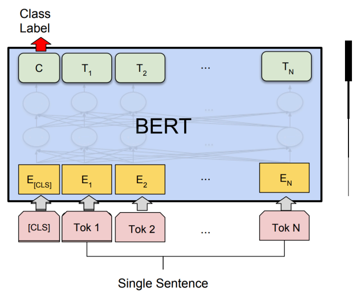
Devlin et al. (2019)

What can BERT NOT do?

- BERT **cannot** generate text (at least not in an obvious way)
  - Can fill in MASK tokens, but can't generate left-to-right (well, you could put MASK at the end repeatedly, but this is slow)
- Masked language models are intended to be used primarily for "analysis" tasks

Fine-tuning BERT

- Fine-tune for 1-3 epochs, batch size 2-32, learning rate 2e-5 - 5e-5



(b) Single Sentence Classification Tasks: SST-2, CoLA

- Large changes to weights up here (particularly in last layer to route the right information to [CLS])
- Smaller changes to weights lower down in the transformer
- Small LR and short fine-tuning schedule mean weights don't change much
- Often requires tricky learning rate schedules ("triangular" learning rates with warmup periods)

BERT results, BERT variants

Evaluation: GLUE

Corpus	|Train|	|Test|	Task	Metrics	Domain
			Single-Sentence Tasks		
CoLA	8.5k	**1k**	acceptability	Matthews corr.	misc.
SST-2	67k	1.8k	sentiment	acc.	movie reviews
			Similarity and Paraphrase Tasks		
MRPC	3.7k	1.7k	paraphrase	acc./F1	news
STS-B	7k	1.4k	sentence similarity	Pearson/Spearman corr.	misc.
QQP	364k	**391k**	paraphrase	acc./F1	social QA questions
			Inference Tasks		
MNLI	393k	**20k**	NLI	matched acc./mismatched acc.	misc.
QNLI	105k	5.4k	QA/NLI	acc.	Wikipedia
RTE	2.5k	3k	NLI	acc.	news, Wikipedia
WNLI	634	**146**	coreference/NLI	acc.	fiction books

Wang et al. (2019)

Results

System	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	**Average**
	392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k	-
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.9	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	88.1	91.3	45.4	80.0	82.3	56.0	75.2
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.1	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**91.1**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**81.9**

- Huge improvements over prior work (even compared to ELMo)
- Effective at "sentence pair" tasks: textual entailment (does sentence A imply sentence B), paraphrase detection

Devlin et al. (2018)

RoBERTa

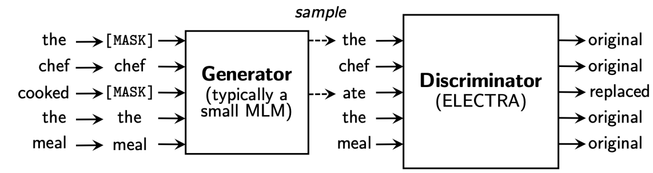
- "Robustly optimized BERT"
- 160GB of data instead of 16 GB
- Dynamic masking: standard BERT uses the same MASK scheme for every epoch, RoBERTa recomputes them
- New training + more data = better performance

Model	data	bsz	steps	SQuAD (v1.1/2.0)	MNLI-m	SST-2
RoBERTa						
with BOOKS + WIKI	16GB	8K	100K	93.6/87.3	89.0	95.3
+ additional data (§3.2)	160GB	8K	100K	94.0/87.7	89.3	95.6
+ pretrain longer	160GB	8K	300K	94.4/88.7	90.0	96.1
+ pretrain even longer	160GB	8K	500K	**94.6/89.4**	**90.2**	**96.4**
BERT$_{LARGE}$						
with BOOKS + WIKI	13GB	256	1M	90.9/81.8	86.6	93.7

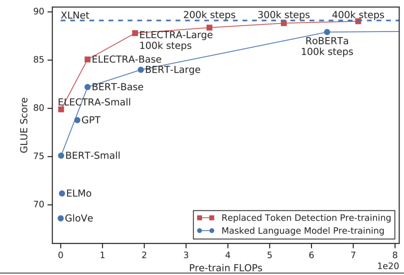
Liu et al. (2019)

ELECTRA

Clark et al. (2020)



- Discriminator to *detect* replaced tokens rather than a generator to actually *predict* what those tokens are
- More efficient, strong performance



DeBERTa

- Slightly better variant

He et al. (2021)

$$\begin{aligned} A_{i,j} &= \{\boldsymbol{H_i}, \boldsymbol{P_{i|j}}\} \times \{\boldsymbol{H_j}, \boldsymbol{P_{j|i}}\}^{\intercal} \\ &= \boldsymbol{H_i}\boldsymbol{H_j}^{\intercal} + \boldsymbol{H_i}\boldsymbol{P_{j|i}}^{\intercal} + \boldsymbol{P_{i|j}}\boldsymbol{H_j}^{\intercal} + \boldsymbol{P_{i|j}}\boldsymbol{P_{j|i}}^{\intercal} \end{aligned} \tag{2}$$

That is, the attention weight of a word pair can be computed as a sum of four attention scores using disentangled matrices on their contents and positions as *content-to-content*, *content-to-position*, *position-to-content*, and *position-to-position* [2].

Model	CoLA Mcc	QQP Acc	MNLI-m/mm Acc	SST-2 Acc	STS-B Corr	QNLI Acc	RTE Acc	MRPC Acc	Avg.
BERT$_{large}$	60.6	91.3	86.6/-	93.2	90.0	92.3	70.4	88.0	84.05
RoBERTa$_{large}$	68.0	92.2	90.2/90.2	96.4	92.4	93.9	86.6	90.9	88.82
XLNet$_{large}$	69.0	92.3	90.8/90.8	**97.0**	92.5	94.9	85.9	90.8	89.15
ELECTRA$_{large}$	69.1	**92.4**	90.9/-	96.9	92.6	95.0	88.0	90.8	89.46
DeBERTa$_{large}$	**70.5**	92.3	**91.1/91.1**	96.8	**92.8**	**95.3**	**88.3**	**91.9**	**90.00**

Using BERT

- HuggingFace Transformers: big open-source library with most pre-trained architectures implemented, weights available
- Lots of standard models…

Model architectures

🤗 Transformers currently provides the following NLU/NLG architectures:

1. **BERT** (from Google) released with the paper BERT: Pre-training of Deep Understanding by Jacob Devlin, Ming-Wei Chang, Kenton Lee and Krist
2. **GPT** (from OpenAI) released with the paper Improving Language Under Radford, Karthik Narasimhan, Tim Salimans and Ilya Sutskever.
3. **GPT-2** (from OpenAI) released with the paper Language Models are Un Jeffrey Wu*, Rewon Child, David Luan, Dario Amodei** and Ilya Sutskev
4. **Transformer-XL** (from Google/CMU) released with the paper Transform Fixed-Length Context by Zihang Dai*, Zhilin Yang*, Yiming Yang, Jaime
5. **XLNet** (from Google/CMU) released with the paper XLNet: Generalized Understanding by Zhilin Yang*, Zihang Dai*, Yiming Yang, Jaime Carbon
6. **XLM** (from Facebook) released together with the paper Cross-lingual L and Alexis Conneau.
7. **RoBERTa** (from Facebook), released together with the paper a Robustly

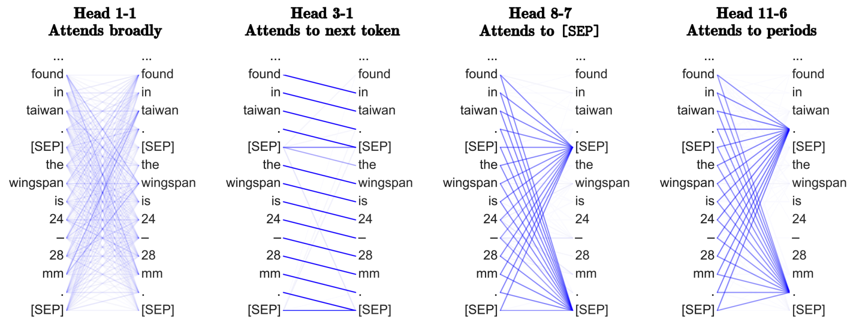
…

and "community models"

- mrm8488/spanbert-large-finetuned-tacred
- mrm8488/xlm-multi-finetuned-xquadv1
- nlpaueb/bert-base-greek-uncased-v1
- nlptown/bert-base-multilingual-uncased-sentiment
- patrickvonplaten/reformer-crime-and-punish
- redewiedergabe/bert-base-historical-german-rw-cased
- roberta-base
- severinsimmler/literary-german-bert
- seyonec/ChemBERTa-zinc-base-v1

…

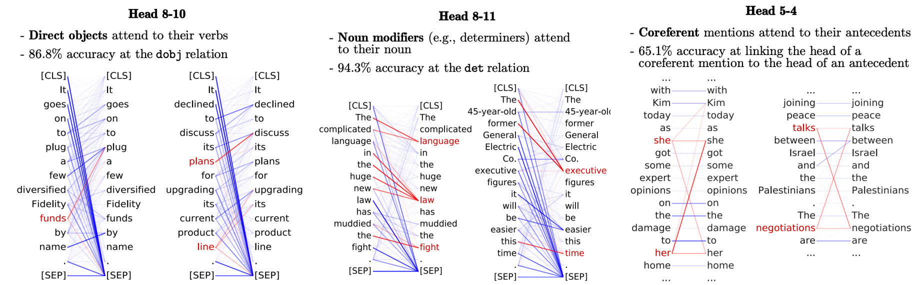
What does BERT learn?



- Heads on transformers learn interesting and diverse things: content heads (attend based on content), positional heads (based on position), etc.

Clark et al. (2019)

What does BERT learn?



- Still way worse than what supervised systems can do, but interesting that this is learned organically

Clark et al. (2019)

Subword Tokenization

Handling Rare Words

- Words are a difficult unit to work with. Why?
  - When you have 100,000+ words, the final matrix multiply and softmax start to dominate the computation, many params, still some words you haven't seen, doesn't take advantage of morphology, …
- Character-level models were explored extensively in 2016-2018 but simply don't work well — becomes very expensive to represent sequences

Subword Tokenization

- Subword tokenization: wide range of schemes that use tokens that are **between characters and words** in terms of granularity
- These "word pieces" may be full words or parts of words

  _the **_eco tax** _port i co _in _Po nt - de - Bu is …

- _ indicates the word piece starting a word (can think of it as the space character).

Sennrich et al. (2016)

Subword Tokenization

- Subword tokenization: wide range of schemes that use tokens that are **between characters and words** in terms of granularity
- These "word pieces" may be full words or parts of words

  _the **_eco tax** _port i co _in _Po nt - de - Bu is …

  Output: _le _port ique **_éco taxe** _de _Pont - de - Bui s

- Can achieve transliteration with this, subword structure makes some translations easier to achieve

Sennrich et al. (2016)

Byte Pair Encoding (BPE)

- Start with every individual byte (basically character) as its own symbol

```
for i in range(num_merges):
  pairs = get_stats(vocab)
  best = max(pairs, key=pairs.get)
  vocab = merge_vocab(best, vocab)
```

- Count bigram character cooccurrences
- Merge the most frequent pair of adjacent characters
- Doing 8k merges => vocabulary of around 8000 word pieces. Includes many whole words
- Most SOTA NMT systems use this on both source + target

Sennrich et al. (2016)

Byte Pair Encoding (BPE)

Original:	furiously
BPE:	_fur | iously
Unigram LM:	_fur | ious | ly

(b)

Original:	tricycles
BPE:	_t | ric | y | cles
Unigram LM:	_tri | cycle | s

Original:	Completely preposterous suggestions
BPE:	_Comple | t | ely | _prep | ost | erous | _suggest | ions
Unigram LM:	_Complete | ly | _pre | post | er | ous | _suggestion | s

- What do you see here?
- BPE produces less linguistically plausible units than another technique based on a unigram language model: rather than greedily merge, find chunks which make the sequence look likely under a unigram LM
- Unigram LM tokenizer leads to slightly better BERT

Bostrom and Durrett (2020)

What's in the token vocab?

Matthew Watkins
@SoC_trilogy

I've just found out that several of the anomalous GPT tokens ("TheNitromeFan", " SolidGoldMagikarp", " davidjl", " Smartstocks", " RandomRedditorWithNo",) are handles of people who are (competitively? collaboratively?) counting to infinity on a Reddit forum. I kid you not.

reddit r/artbn_bots Search Reddit

- Full list.
- Top 1000
- Top 500
- Top 250
- Top 100

Rank	User	Counts
1	/u/davidjl123	163477
2	/u/Smartstocks	113829
3	/u/atomicimploder	103178
4	/u/TheNitromeFan	84581
5	/u/SolidGoldMagikarp	65753
6	/u/RandomRedditorWithNo	63434
7	/u/rideride	59024
8	/u/Mooraell	57785
9	/u/Removedpixel	55080
10	/u/Adinida	48415
11	/u/rschaosid	47132

Tokenization Today

- **All pre-trained** models use some kind of subword tokenization with a tuned vocabulary; usually between 50k and 250k pieces (larger number of pieces for multilingual models)
- As a result, classical word embeddings like GloVe **are not used**. All subword representations are randomly initialized and learned in the Transformer models

Takeaways

- Pre-trained models and BERT are very powerful for a range of NLP tasks
- These models have enabled big advances in NLI and QA specifically
- Next time: pre-trained decoders (GPT-3) and encoder-decoder models (T5)