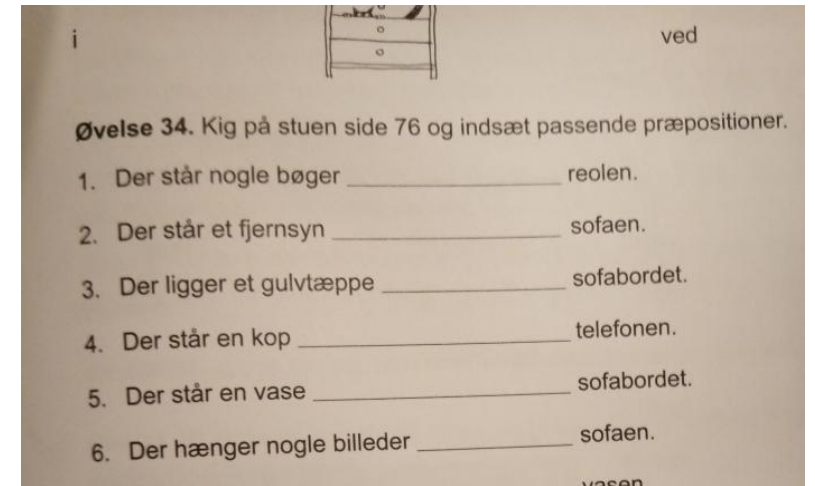




Transformers, BERT and transfer learning with language models

Rob van der Goot



Today

- Note on UNK label
- BERT
- Transformers (only relevant part)
- (Easy) transfer learning
- Questions for Dirk Hovy
- HPC

Note on UNK label

In Viterbi and BERT assignment, UNK is included as label in provided code.

- Not standard
- Makes things easier for unseen labels in dev/test data:

```
# Data reading with UNK
for line in open(sys.argv[1]):
    tok = line.strip().split('\t')
    label = tok[1]
    if label not in label2id:
        label = label2id['UNK']
    ...

# Data reading without UNK
for line in open(sys.argv[1]):
    tok = line.strip().split('\t')
    label = tok[1]
    if label not in label2id:
        label2id[label] = len(label2id)
        id2label.append(label)
        label = label2id[label]
    ...
```

```
# Loss with UNK
```

```
loss_function = torch.nn.CrossEntropyLoss()  
loss = loss_function(logits, gold)
```

```
# Loss without UNK
```

```
loss_function = torch.nn.CrossEntropyLoss()  
gold = gold[gold >= len(label2id)] = ?  
loss = loss_function(logits, gold)
```

- https://www.youtube.com/watch?v=K_8vCovrmag

BERT: Bidirectional Encoder Representations from Transformers

Parts of BERT:

- Tokenization
- Input embeddings
- Encoder (transformers)
- Masked Language Modeling
- Fine-tuning

Use of BERT:

- Transfer learning with BERT

Tokenization

- Term is re-defined!
- BERT uses four steps
 - NFD Normalization
 - Separate punctuation and convert whitespaces
 - Subword segmentation
 - Add special tokens

Normalization

- Normalization Form Canonical Decomposition (NFD)
- Extract accents from characters (and remove separate accents!)
- Be careful!

```
>>> tokenizer = AutoTokenizer.from_pretrained('bert-base-multilingual-cased')
>>> tokenizer.decode(tokenizer.encode('If you do not see yourselves then'))
"[CLS] If you don't see yourselves then [SEP]"
>>>
>>> tokenizer = AutoTokenizer.from_pretrained('bert-base-multilingual-cased', use_fast=False)
>>> tokenizer.decode(tokenizer.encode('If you do not see yourselves then'))
'[CLS] If you do not see yourselves then [SEP]'
```


Tokenization in BERT

```
def _is_punctuation(char):  
    """Checks whether `char` is a punctuation character."""  
    cp = ord(char)  
    # We treat all non-letter/number ASCII as punctuation.  
    # Characters such as "^", "$", and "`" are not in the Unicode  
    # Punctuation class but we treat them as punctuation anyways, for  
    # consistency.  
    if (cp >= 33 and cp <= 47) or (cp >= 58 and cp <= 64) or (cp >= 91 and cp <= 96) or (cp >= 123 and cp <= 126):  
        return True  
    cat = unicodedata.category(char)  
    if cat.startswith("P"):  
        return True  
    return False
```

Subword segmentation in BERT

- Start with a vocabulary of all characters as subwords, prefix non-start characters with ##
- Rank all possible merges with score:

$$score_{subw1, subw2} = \frac{count(subw1, subw2)}{count(subw1) * count(subw2)}$$

- Apply the highest scoring merge
- Redo until desired vocabulary size

Original subword algorithm

1. Initialize the word unit inventory with the basic Unicode characters (Kanji, Hiragana, Katakana for Japanese, Hangul for Korean) and including all ASCII, ending up with about 22000 total for Japanese and 11000 for Korean.
2. Build a language model on the training data using the inventory from 1.
3. Generate a new word unit by combining two units out of the current word inventory to increment the word unit inventory by one. Choose the new word unit out of all possible ones that increases the likelihood on the training data the most when added to the model.
4. Goto 2 until a predefined limit of word units is reached or the likelihood increase falls below a certain threshold.

From: Japanese and Korean Voice Search (M. Schuster and K. Nakajima, 2012)

Subword segmentation in BERT

To apply:

- Greedily match longest possible subwords from left
- Use [UNK] for unseen character sequences

Subword segmentation in BERT

- English BERT: ~30,000 subwords
- Multi-lingual BERT (over 100 languages): ~120,000 subwords
- English spell checker word list: ~127,000

en_tok: ['We', 'recommend', 'these', 'cabins', '!']

ml_tok: ['We', 're', '##com', '##mend', 'these', 'cabin', '##s', '!']

- [Rust et al \(2021\)](#) evaluate the effect of this difference

What is a subword?

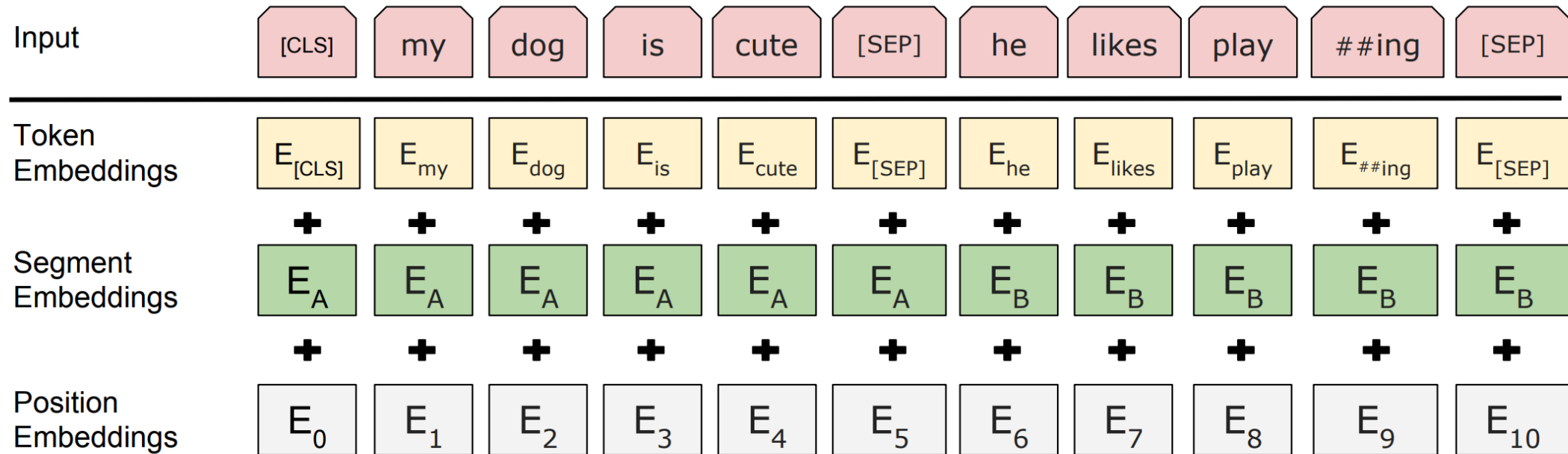
- Syllable: a unit of pronunciation
- Morpheme: smallest meaningful constituent of a linguistic expression
- Tokenizer -> tok-en-iz-er or token-izer

Special tokens in BERT

- Start with [CLS], end with [SEP]
- For two-sentence inputs, include [SEP] in the middle
- Note that input to model are indices:

[101, 1284, 18029, 1292, 23108, 106, 102]

Input embeddings in BERT




Input embeddings in BERT

- Label -> vector of size 768
- Segment: 2 labels
- Position: 512 labels
- Original transformers use Sinusoidal function for generating positional embeddings
- In BERT all embeddings are learned

BERT Encoder

- 12 transformer (encoding) layers:
- [Attention is all you need. Vaswani et al \(2017\)](#)
- Attention: learn what to attend to. In ELMO: layer weights

Transformer

- Transforms input to output 
- Based on self-attention
- Originally proposed for machine translation

Terminology

- All input available: encoder
- Left input available: decoder (note that figure 9.15 in the book is a decoder model, we will focus on encoder models only)

Self-attention

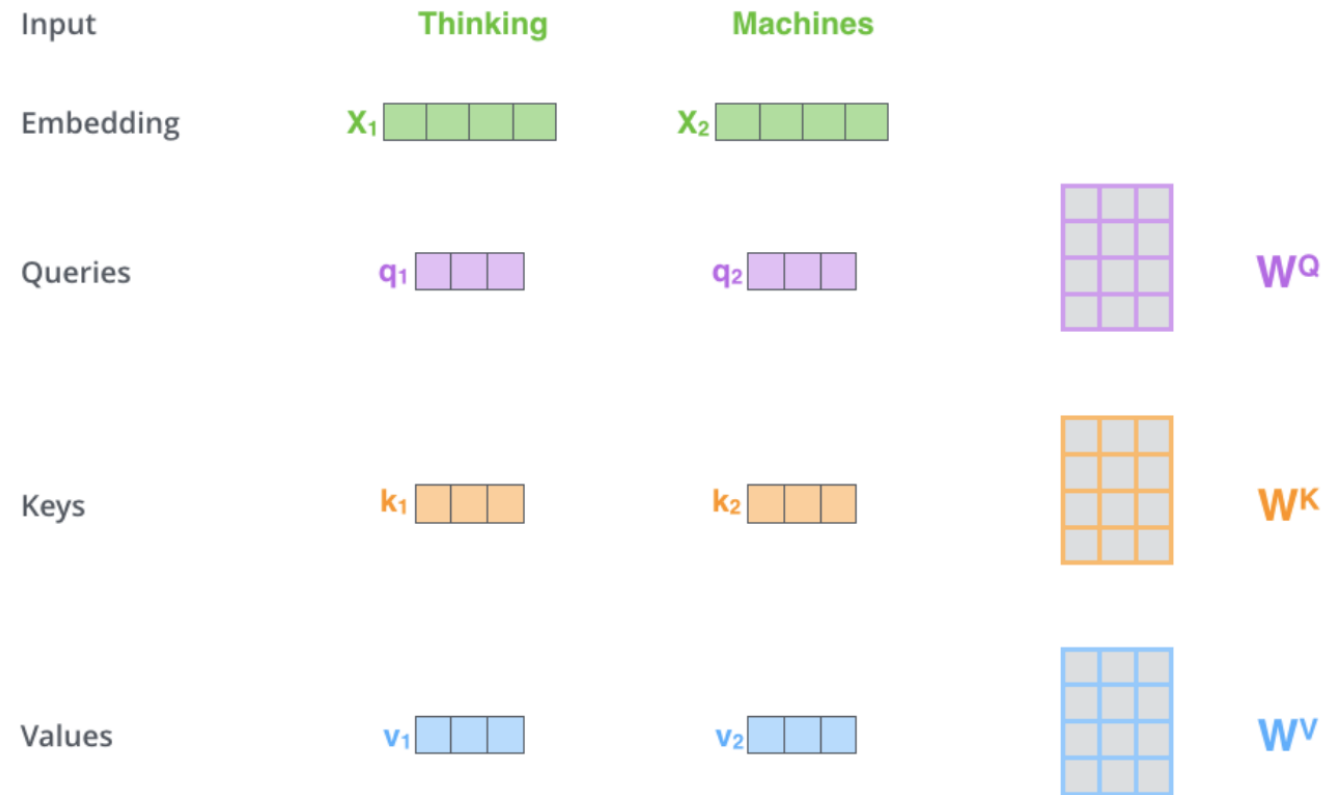
- Which information from input to attend to?

Each embeddings has three functions:

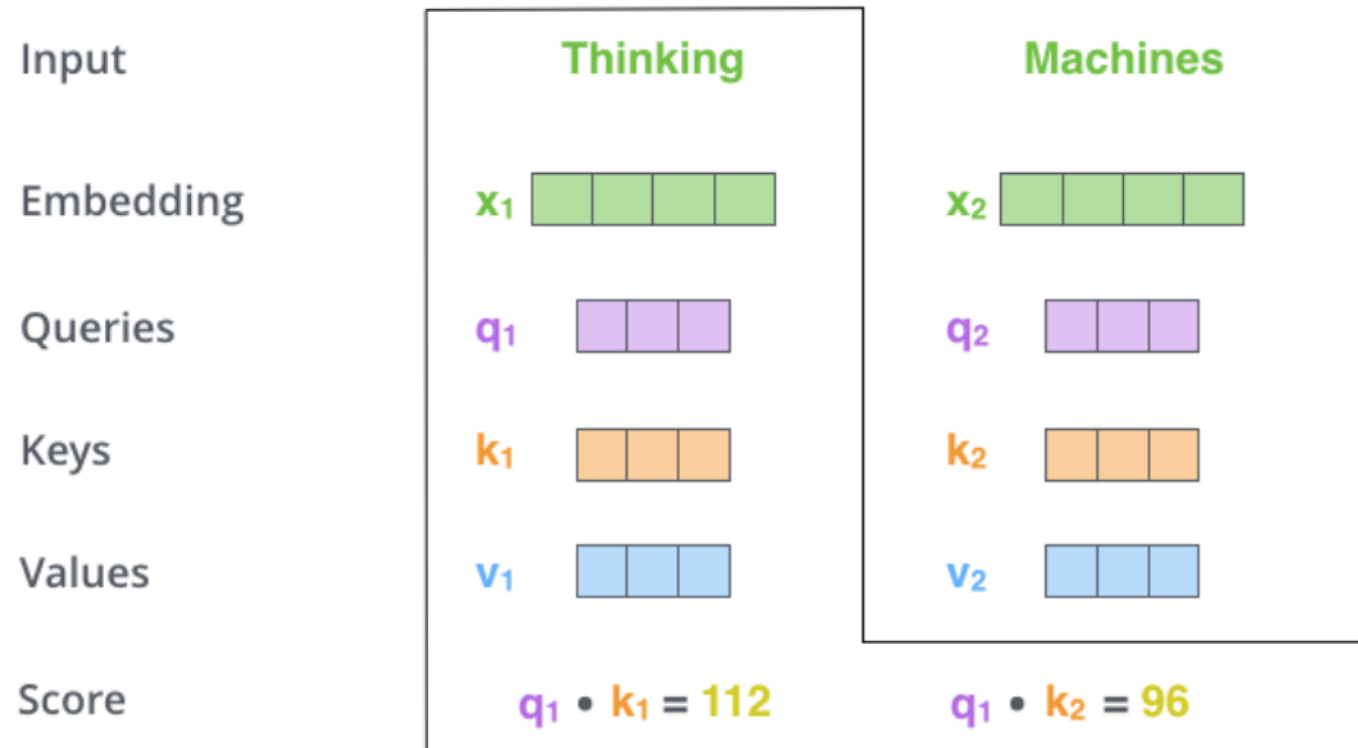
- Query: Learn what to focus on from current word
- Key: How important as context
- Value: Representation of subword (used as output)

These are obtained by the dot product of the word embedding with a weight matrix ($\sim 1024 \times 1024$).

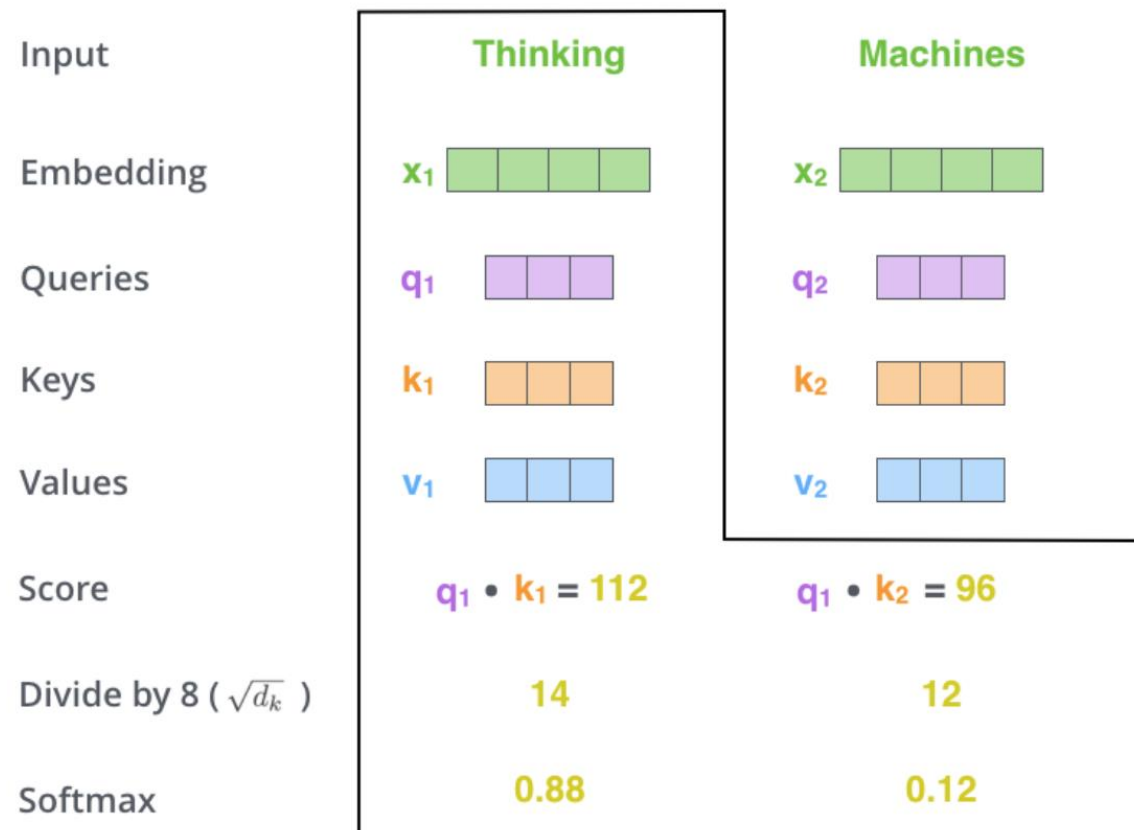
$$\mathbf{q}_i = \mathbf{W}^Q \mathbf{x}_i; \quad \mathbf{k}_i = \mathbf{W}^K \mathbf{x}_i; \quad \mathbf{v}_i = \mathbf{W}^V \mathbf{x}_i \quad (9.34)$$



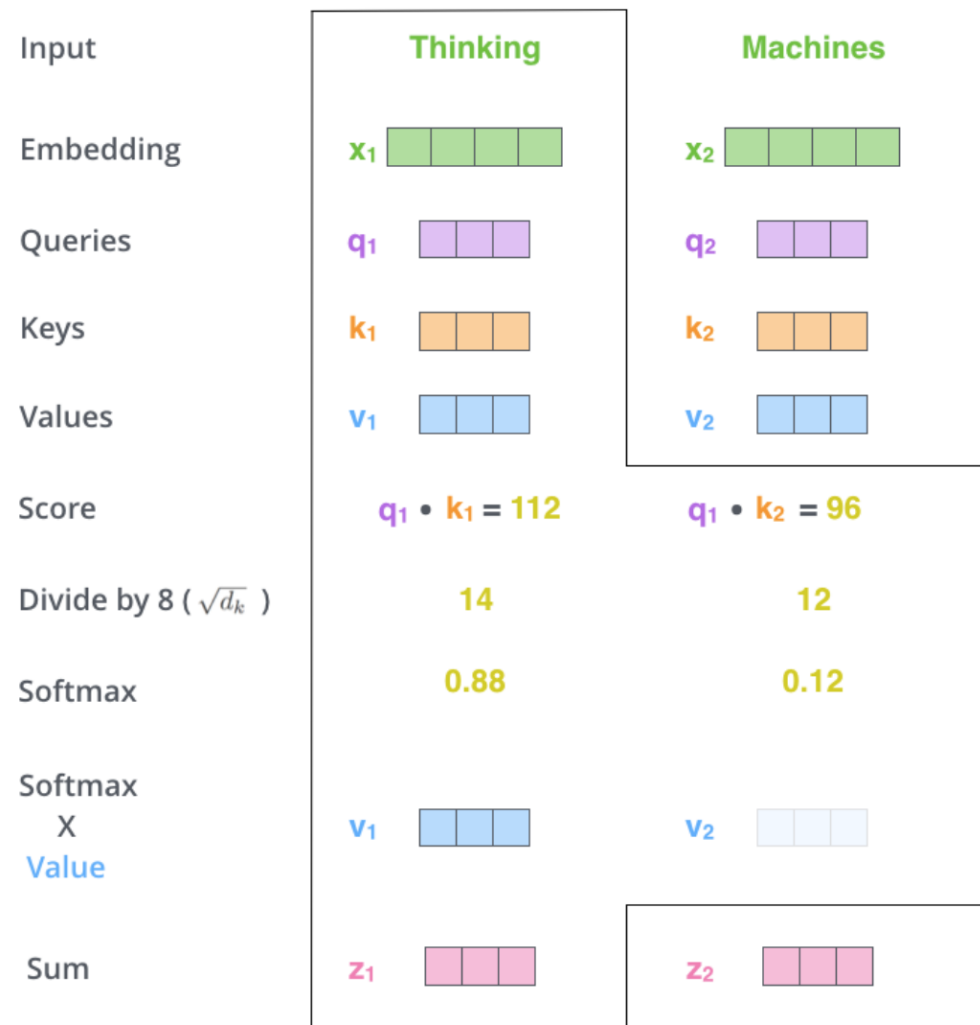
- Images from: <https://jalammar.github.io/illustrated-transformer/>



- Images from: <https://jalammar.github.io/illustrated-transformer/>



- Images from: <https://jalammar.github.io/illustrated-transformer/>



- Images from: <https://jalammar.github.io/illustrated-transformer/>

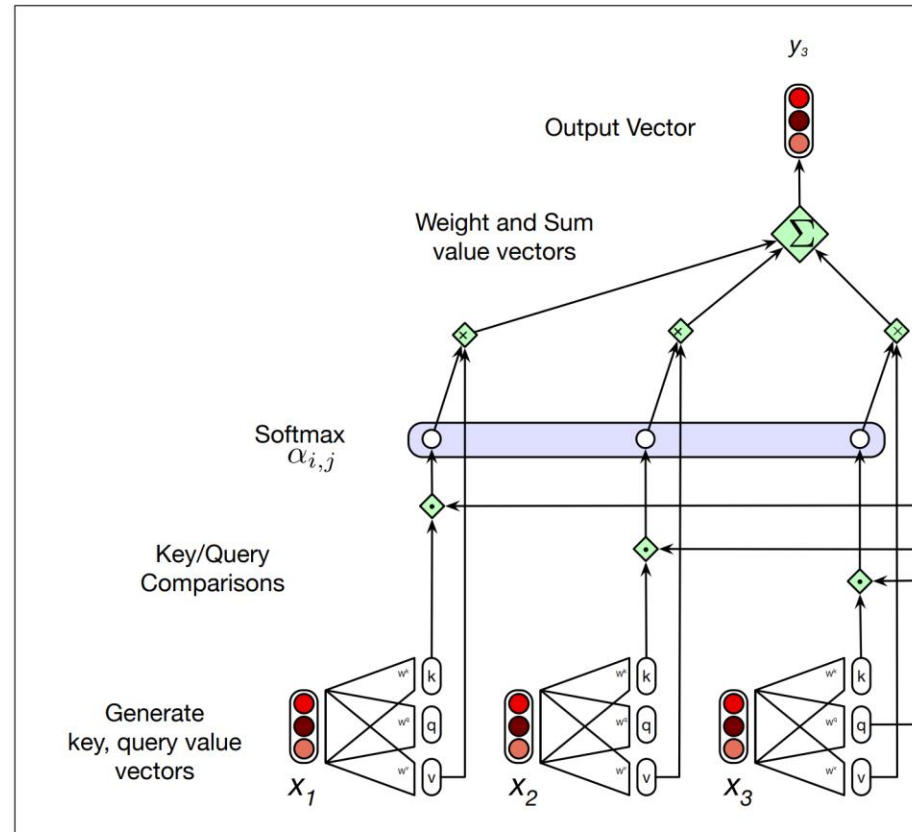


Figure 9.16 Calculating the value of y_3 , the third element of a sequence using causal (left-to-right) self-attention.

Is that all (we need)?

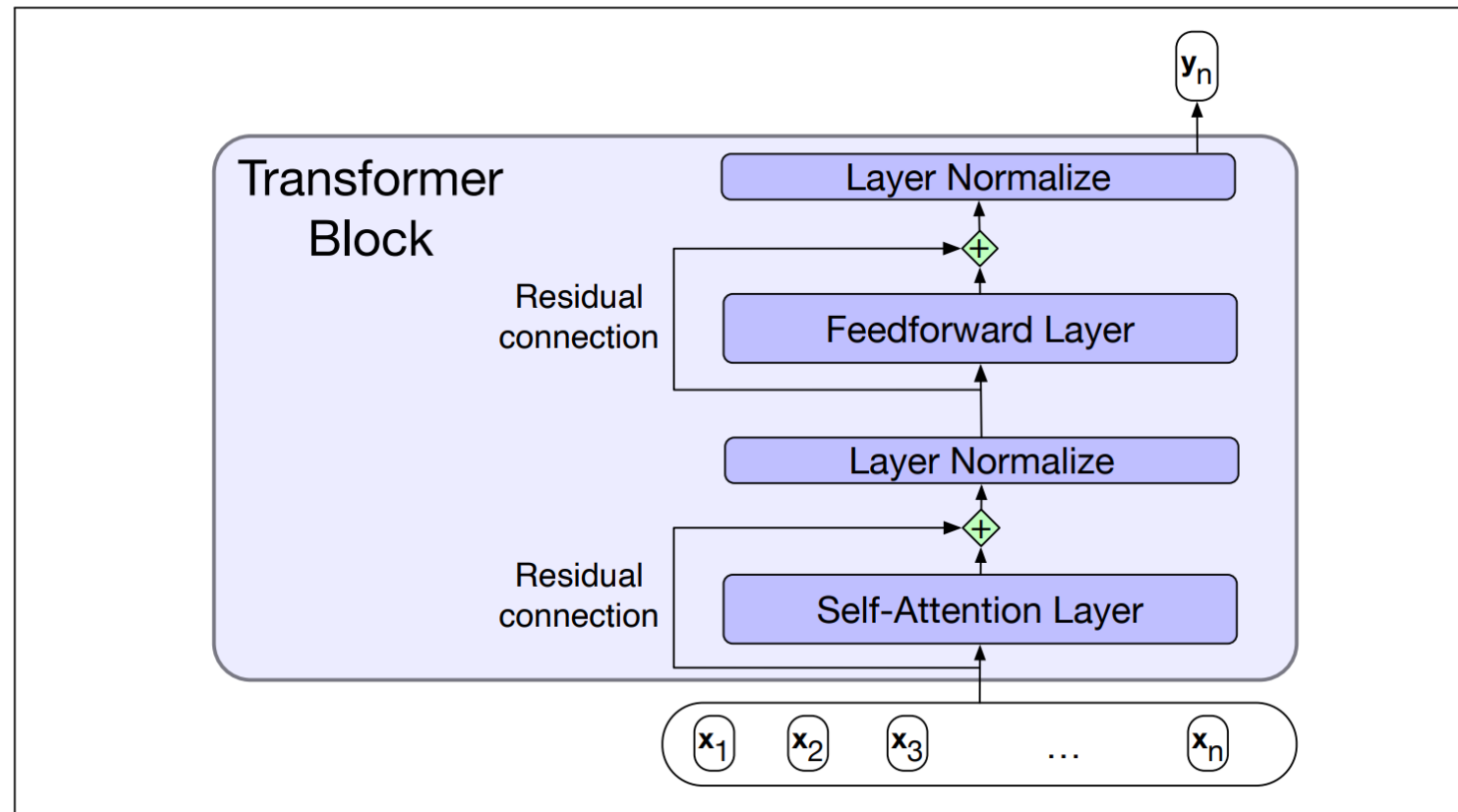


Figure 9.18 A transformer block showing all the layers.

Multihead Self-attention

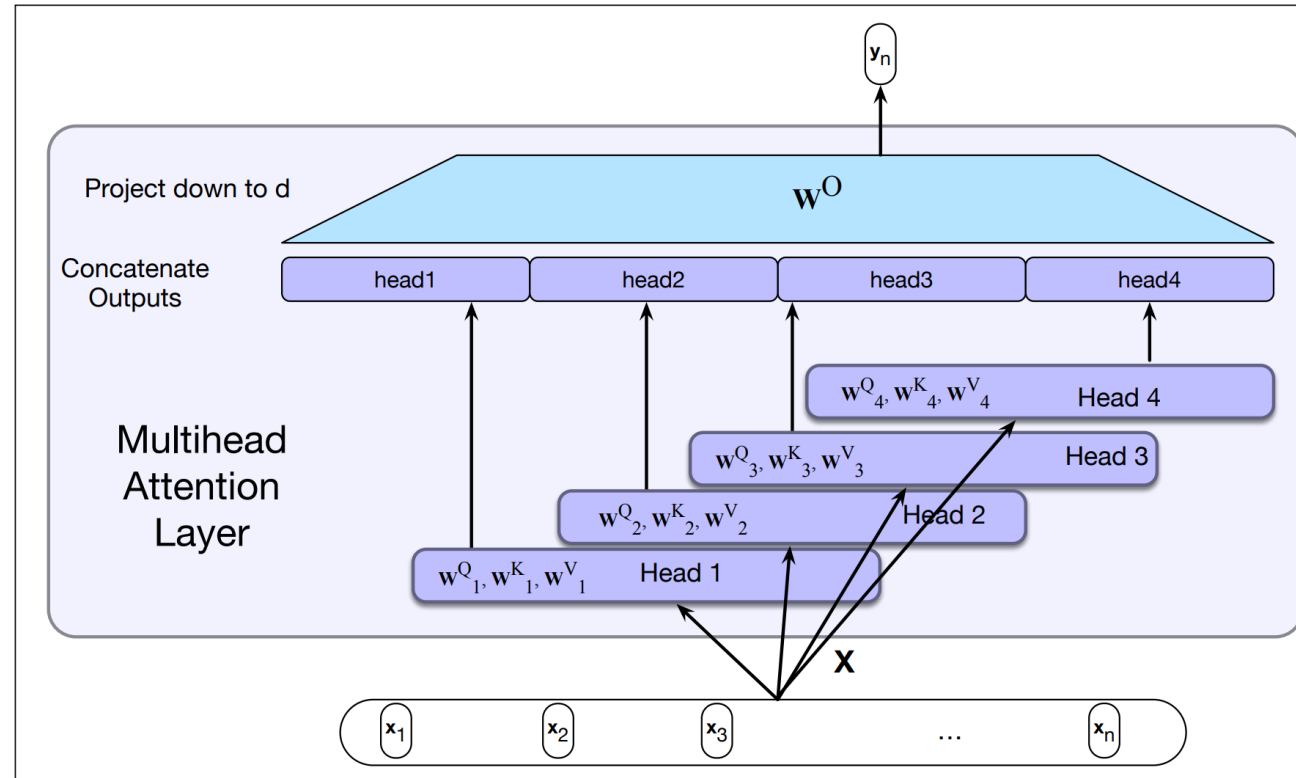


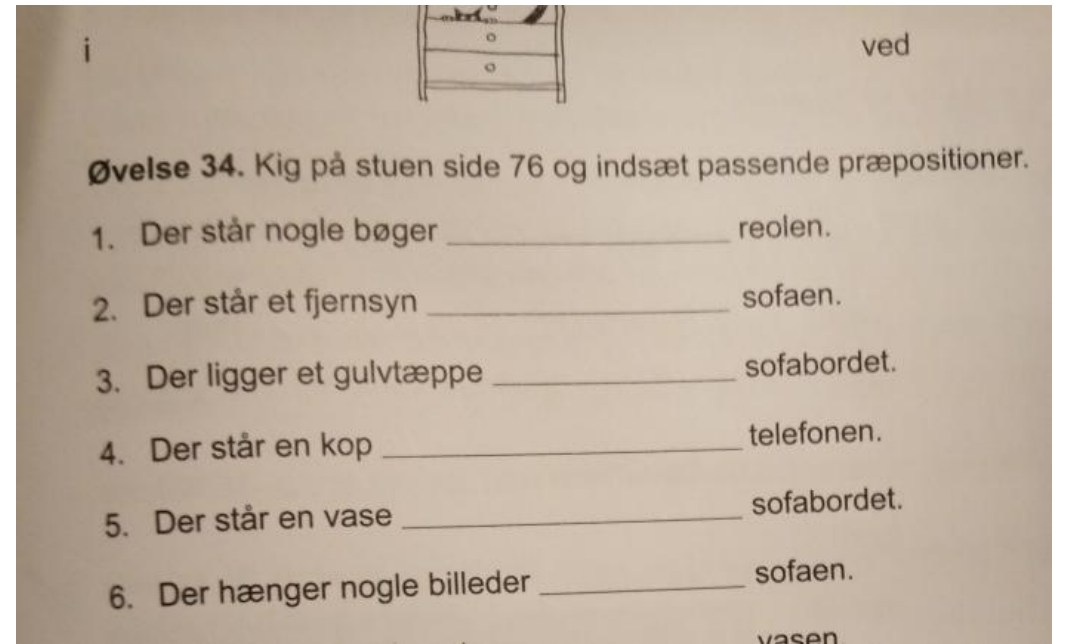
Figure 9.19 Multihead self-attention: Each of the multihead self-attention layers is provided with its own set of key, query and value weight matrices. The outputs from each of the layers are concatenated and then projected down to d , thus producing an output of the same size as the input so layers can be stacked.

Self-attention as analysis

- <https://huggingface.co/exbert/?model=xlm-roberta-base>
- But: [Attention is not Explanation](#)
- But: [Attention is not not Explanation](#)

Training objective

- Masked Language Modeling (MLM)
 - 15% masked
 - 80% [MASK]
 - 10% random
 - 10% self
- Next Sentence Prediction (NSP)
 - 50% isNext, 50% notNext



Hyperparameters BERT-base

- Size of all embeddings: 768
- 12 transformer layers, with 12 attention heads each
- Max length: 512
- Vocab size: ~30,000
- Masking: 15%
- 1,000,000 training steps
- 128,000 tokens/batch
- Wikipedia/books data
- More in Appendix A.2

Fine-tuning

- Remove subword prediction layer, add a new feedforward layer for prediction of our target task:
- Embedding of [CLS] for sentences
- Embedding of first/last subword of token for token-level tasks
- Note that it is now common to always update the whole network (as opposed to ELMO-times)

Fine-tuning

System	Dev F1	Test F1
ELMo (Peters et al., 2018a)	95.7	92.2
CVT (Clark et al., 2018)	-	92.6
CSE (Akbiik et al., 2018)	-	93.1
Fine-tuning approach		
BERT _{LARGE}	96.6	92.8
BERT _{BASE}	96.4	92.4
Feature-based approach (BERT _{BASE})		
Embeddings	91.0	-
Second-to-Last Hidden	95.6	-
Last Hidden	94.9	-
Weighted Sum Last Four Hidden	95.9	-
Concat Last Four Hidden	96.1	-
Weighted Sum All 12 Layers	95.5	-

Table 7: CoNLL-2003 Named Entity Recognition results. Hyperparameters were selected using the Dev set. The reported Dev and Test scores are averaged over 5 random restarts using those hyperparameters.

Results BERT

GLUE: General Language Understanding Evaluation

- Linguistic acceptability
- Sentiment analysis
- Paraphrase detection
- Textual similarity (regression)
- Recognizing textual entailment (*4)

Recognizing textual entailment

Entail:

- The cup is shiny The cup glitters
- a man breakdances A man dances (NOTE: directed)

Contradict:

- The sky is blue It is night
- dog barking An animal is asleep.

Neutral:

- A woman skiing A sad woman skiing (NOTE: directed)
- a ski resort the resort is good

Results BERT

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	Average -
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	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
BERT _{BASE}	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
BERT _{LARGE}	86.7/85.9	72.1	92.7	94.9	60.5	86.5	89.3	70.1	82.1

Transfer learning with mBERT

- In book: pre-training (wikipedia) then fine-tuning (target dataset)
- But, you can transfer to a language variety through language modeling (without annotated data)

mBERT

- No publication, only README: <https://github.com/google-research/bert/blob/master/multilingual.md>
- Train on data of ~100 languages
- Remarkable cross-lingual performance when fine-tuning on only one language

xSID

Rob van der Goot, Ibrahim Sharaf, Aizhan Imankulova, Ahmet Üstün, Marija Stepanović, Alan Ramponi, Siti Oryza Khairunnisa, Mamoru Komachi and Barbara Plank



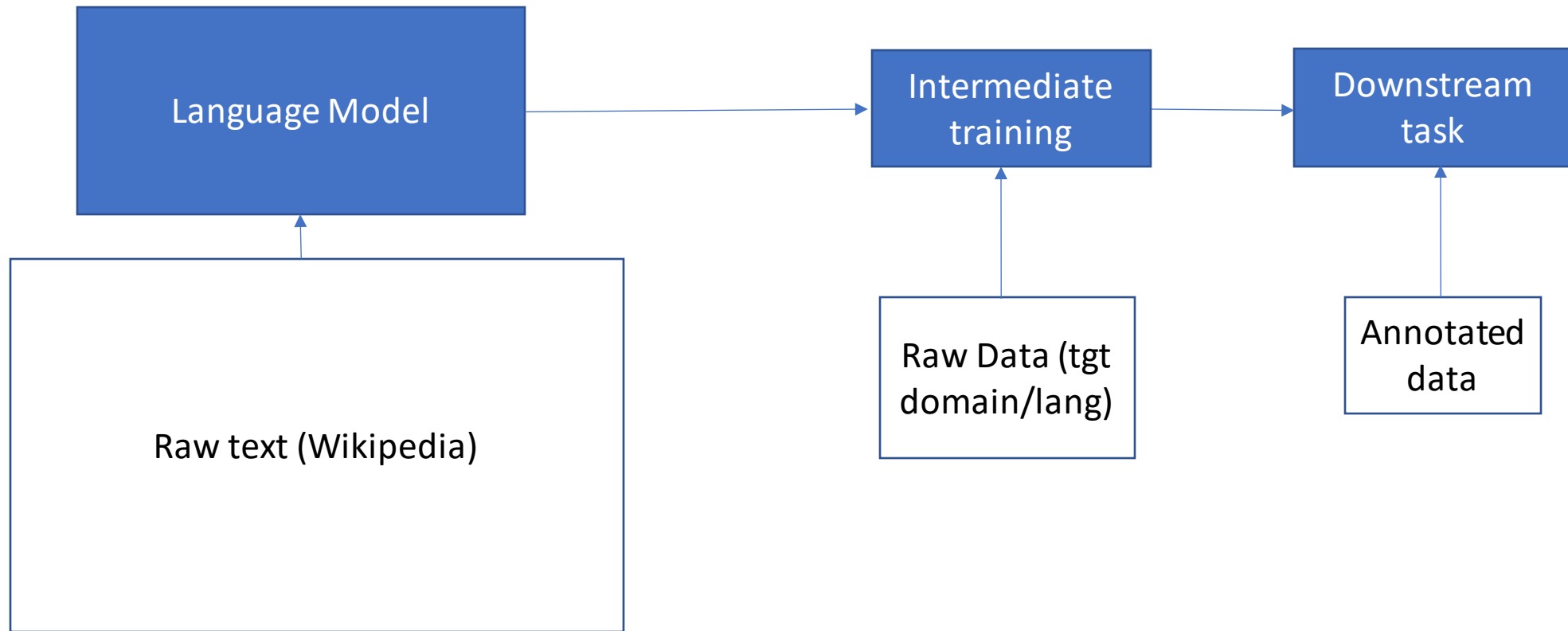
Lang.	Language Family	Annotation
ar	Afro-Asiatic	أود أن أرى مواعيد عرض فيلم Silly Movie 2.0 في دار السينما
da	Indo-European	Jeg vil gerne se spilletiderne for Silly Movie 2.0 i biografen
de	Indo-European	Ich würde gerne den Vorstellungsbeginn für Silly Movie 2.0 im Kino sehen
de-st	Indo-European	I mecht es Programm fir Silly Movie 2.0 in Film Haus sechn
en	Indo-European	I'd like to see the showtimes for Silly Movie 2.0 at the movie house
id	Austronesian	Saya ingin melihat jam tayang untuk Silly Movie 2.0 di gedung bioskop
it	Indo-European	Mi piacerebbe vedere gli orari degli spettacoli per Silly Movie 2.0 al cinema
ja	Japonic	映画館の Silly Movie 2.0 の上映時間を見せて。
kk	Turkic	Мен Silly Movie 2.0 бағдарламасының кинотеатрда көрсетілім уақытын
nl	Indo-European	Ik wil graag de speeltijden van Silly Movie 2.0 in het filmhuis zien
sr	Indo-European	Želela bih da vidim raspored prikazivanja za Silly Movie 2.0 u bioskopu
tr	Turkic	Silly Movie 2.0 'ın sinema salonundaki seanslarını görmek istiyorum
zh	Sino-Tibetan	我想看 Silly Movie 2.0 在影院的放映

xSID

mBERT lang2vec	en	de-st	de	da	nl	it	sr	id	ar	zh	kk	tr	ja*	Avg.
Slots														
base	97.6	48.5	33.0	73.9	80.4	75.0	67.4	71.1	45.8	72.9	48.5	55.7	59.9	61.0
Intents														
base	99.7	67.8	74.2	87.5	72.3	81.7	75.7	80.7	63.1	83.3	60.1	74.7	53.9	72.9

Re-training

If language mode in target language/domain is not available, re-training is a "cheap" alternative.



Types of transformer-based language models

- Autoencoder: masking
- Auto-regressive: seq2seq

Other extensions of BERT

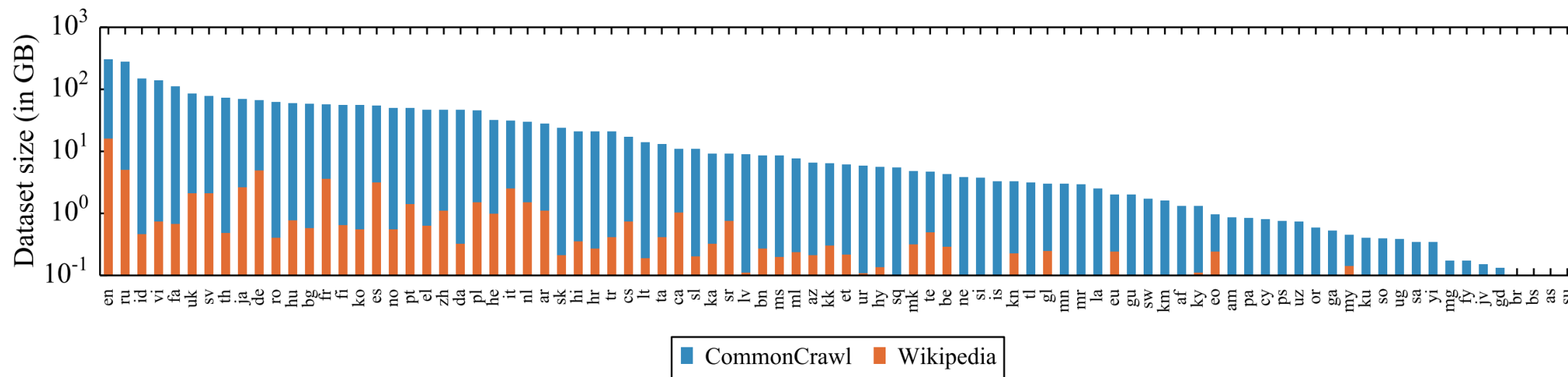
- Hard to keep up!
- Many language specific models
- Multi-lingual auto-encoder models:
- ["microsoft/mdeberta-v3-base", "studio-ousia/mluke-large", "google/rembert", "cardiffnlp/twitter-xlm-roberta-base", "xlm-roberta-large", "bert-base-multilingual-cased", "xlm-roberta-base", 'distilbert-base-multilingual-cased', 'facebook/xlm-roberta-xl', 'microsoft/inoxlm-large', 'bert-base-multilingual-uncased', 'Peltarion/xlm-roberta-longformer-base-4096', 'Peltarion/xlm-roberta-longformer-base-4096', 'studio-ousia/mluke-base', 'facebook/xlm-roberta-xxl', 'xlm-mlm-100-1280']

RoBERTa: An optimized method for pretraining self-supervised NLP systems

- No NSP
- Larger mini-batches and learning rates
- More data (16gb->160gb)

XLM-R

- [Conneau et al. \(2020\)](#)
- 2.5tb training data



XLM-R

- Be careful again:

```
>>> tokenizer = AutoTokenizer.from_pretrained('xlm-roberta-large')
>>> tokenizer.tokenize('² ₣ cm m² km °')
['_2', '_', '₣', '_cm', '_m', '2', '_km', '_°']
```

XLM-R

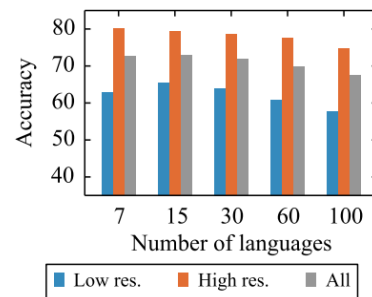


Figure 2: The transfer-interference trade-off: Low-resource languages benefit from scaling to more languages, until dilution (interference) kicks in and degrades overall performance.

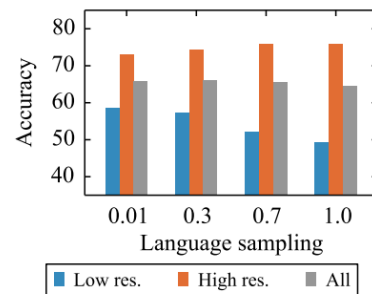


Figure 5: On the high-resource versus low-resource trade-off: impact of batch language sampling for XLM-100.

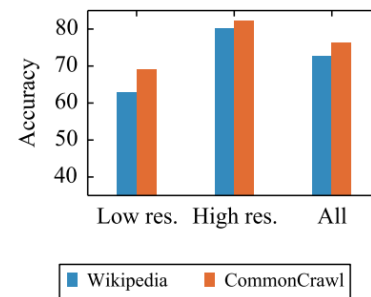


Figure 3: Wikipedia versus CommonCrawl: An XLM-7 obtains significantly better performance when trained on CC, in particular on low-resource languages.

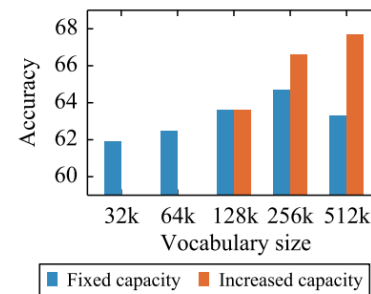


Figure 6: On the impact of vocabulary size at fixed capacity and with increasing capacity for XLM-100.

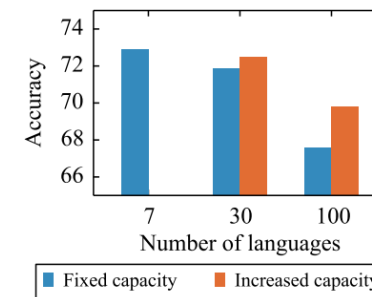


Figure 4: Adding more capacity to the model alleviates the curse of multilinguality, but remains an issue for models of moderate size.

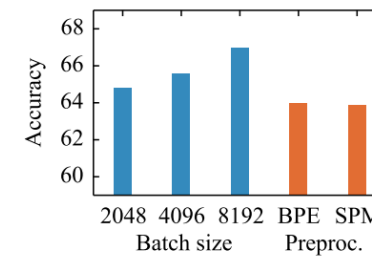


Figure 7: On the impact of large-scale training, and preprocessing simplification from BPE with tokenization to SPM on raw text data.

XLM-R

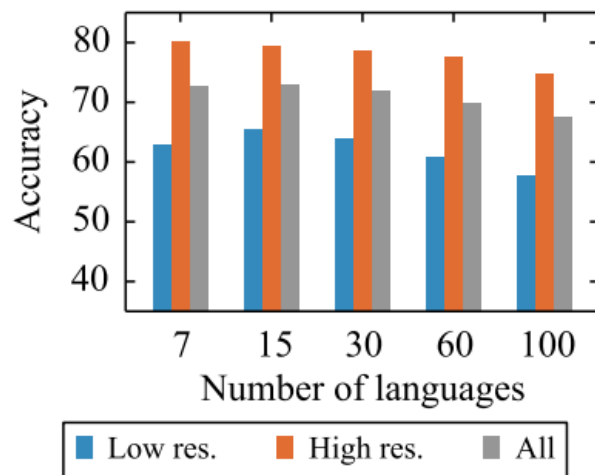


Figure 2: The transfer-interference trade-off: Low-resource languages benefit from scaling to more languages, until dilution (interference) kicks in and degrades overall performance.

XLM-R

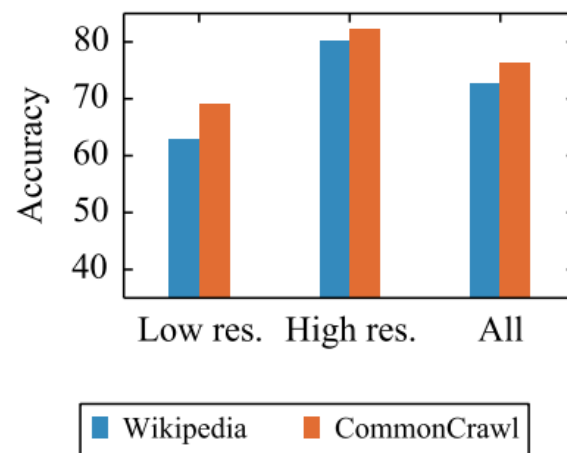


Figure 3: Wikipedia versus CommonCrawl: An XLM-7 obtains significantly better performance when trained on CC, in particular on low-resource languages.

XLM-R

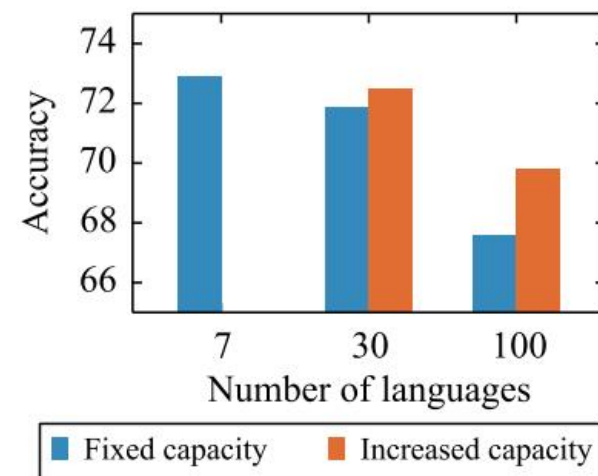


Figure 4: Adding more capacity to the model alleviates the curse of multilinguality, but remains an issue for models of moderate size.

XLM-R

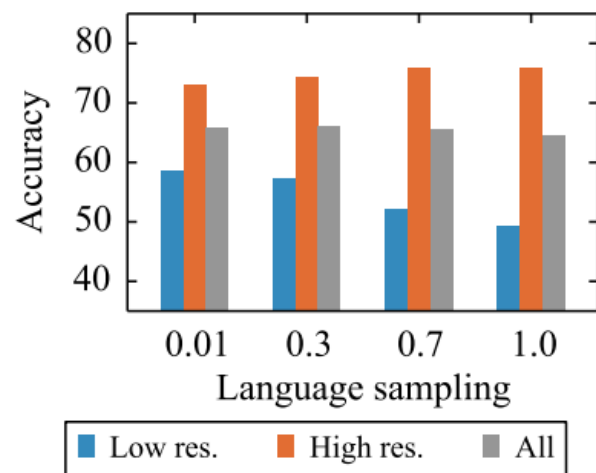


Figure 5: On the high-resource versus low-resource trade-off: impact of batch language sampling for XLM-100.

XLM-R

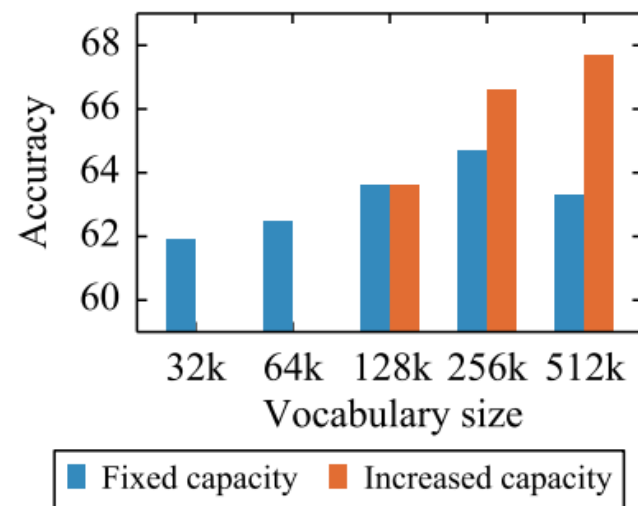


Figure 6: On the impact of vocabulary size at fixed capacity and with increasing capacity for XLM-100.

XLM-R

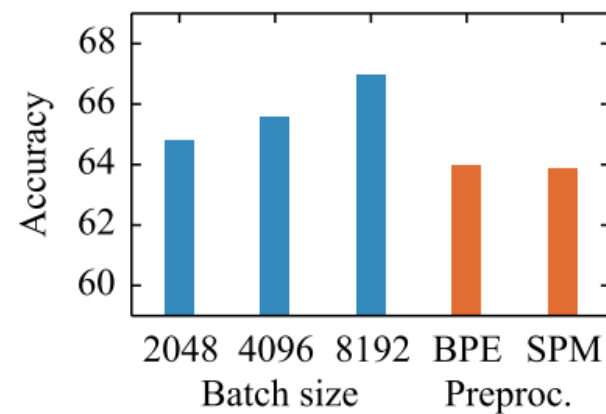


Figure 7: On the impact of large-scale training, and preprocessing simplification from BPE with tokenization to SPM on raw text data.

XLM-R

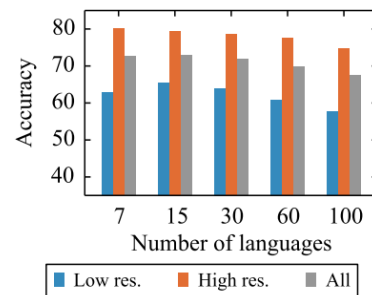


Figure 2: The transfer-interference trade-off: Low-resource languages benefit from scaling to more languages, until dilution (interference) kicks in and degrades overall performance.

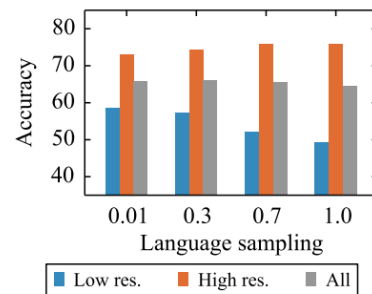


Figure 5: On the high-resource versus low-resource trade-off: impact of batch language sampling for XLM-100.

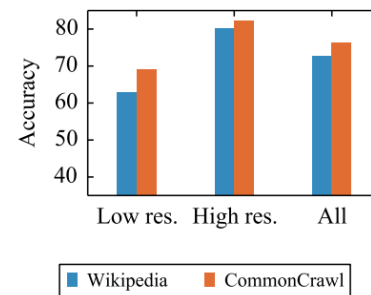


Figure 3: Wikipedia versus CommonCrawl: An XLM-7 obtains significantly better performance when trained on CC, in particular on low-resource languages.

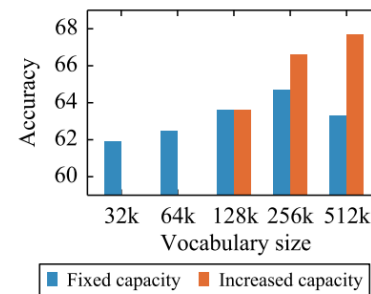


Figure 6: On the impact of vocabulary size at fixed capacity and with increasing capacity for XLM-100.

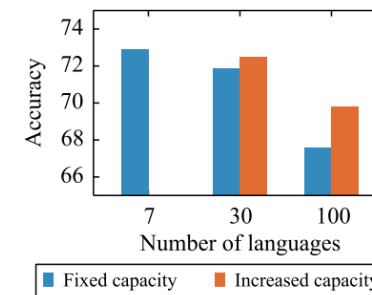


Figure 4: Adding more capacity to the model alleviates the curse of multilinguality, but remains an issue for models of moderate size.

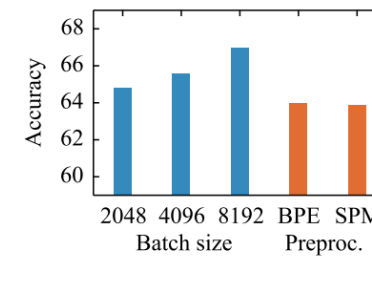


Figure 7: On the impact of large-scale training, and preprocessing simplification from BPE with tokenization to SPM on raw text data.

- Note that XLM-R still performs well across many datasets. Why?

Questions

- How to handle longer inputs?
- Where do the gains come from (compared to w2v/elmo)?
 - Wordpieces
 - Depth
 - Size
 - Transformers
 - Whole input at once
 - Training data
- How is the tokenizer (of mBERT) trained?
- For mBERT, are both sentences from the same language?
- Cased/uncased models?

Questions for Dirk

Second Year Project (Introduction to Natural Language Processing and Deep Learning) (Spring 2023)

 News

 Ordinary exam assignment

Hidden from students

 Re-exam assignment

Hidden from students

 Examination Syllabus

*This activity lists the mandatory literature for the examination.
The examination syllabus is published before the last lecture or earlier.*

 Slack channel

 Repository with slides and assignments

 Speech and Language Processing (SLP)

 Speech and Language Processing (1 pdf)

 Early evaluation

 Questions for Dirk