Large Language Models

Part 1: Introduction & subword tokenization

David Samuel

Language Technology Group, University of Oslo



Table of contents

1. Introduction

2. Subword tokenization

3. References

We are building upon concepts introduced in previous lectures

- Contextualized word embeddings

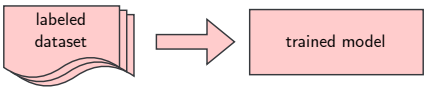
We are building upon concepts introduced in previous lectures

- Contextualized word embeddings
- Representation learning

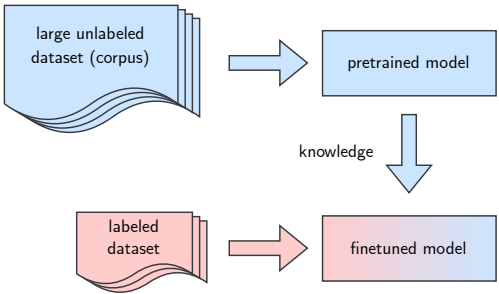
We are building upon concepts introduced in previous lectures

- Contextualized word embeddings
- Representation learning
- Transfer learning

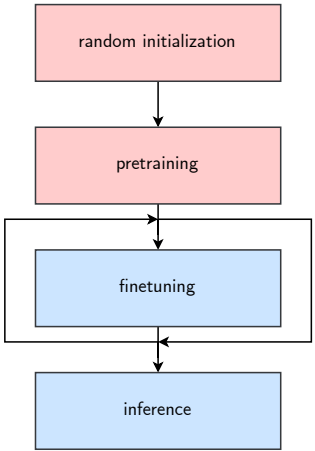
Traditional machine learning



Transfer learning

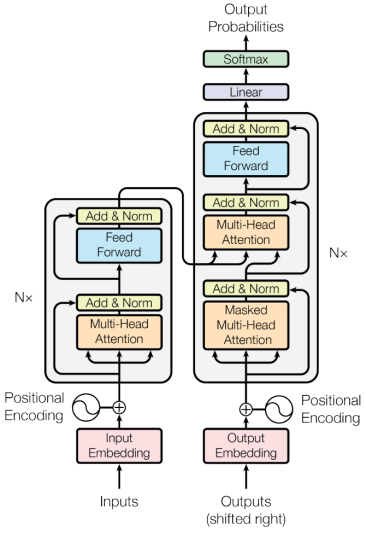


The life of a language model



We are building upon concepts introduced in previous lectures

- Contextualized word embeddings
- Representation learning
- Transfer learning
- Transformers
  - **encoder**: *encodes* the source text
  - **decoder**: takes the encoded text and *decodes* the target text token by token


Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)

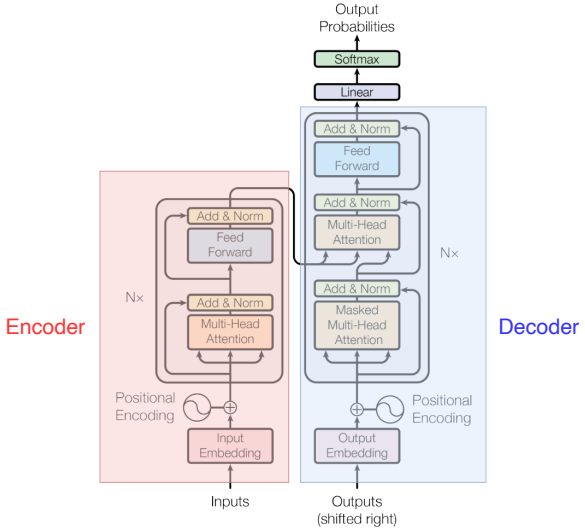

Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Positional Encoding
Output Embedding
Outputs (shifted right)
Encoder
Decoder
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Positional Encoding
Input Embedding
Inputs

Table of contents

1. Introduction

2. Subword tokenization

3. References

Subword tokenization

- Tokenization = *"splitting a text sequence into more digestible chunks"*
- We have already seen word tokenization
  - But there is a near-infinite amount of possible word forms
  - Out-of-vocabulary (OOV) words are frequent even for static word embeddings with millions of word-tokens in their vocabularies
  - word tokens $\rightarrow$ large vocabulary and frequent OOV misses
- Another possibility is character tokenization or even byte tokenization
  - This can solve the problems of tokenizing by words
  - But we will end up with long sequences of tokens, which are very expensive to process

Subword tokenization

- To overcome these issues, neural machine translation has popularized the use of subword tokenization
  - subword = *"a cleverly chosen word or part of a word"*
- Subwords require vocabularies with only 10 000s of tokens without introducing any out-of-vocabulary errors
- $\rightarrow$ Word tokens are no longer used in modern language models, every relevant model relies on subwords now

Examples

- BERT: “*Ex ##ample of token ##ization for IN ##55 ##50 .*”
- XLM-R: “*▁Exam ple ▁of ▁to ken ization ▁for ▁IN 55 50 .*”
- ChatGPT: “*Example Ġof Ġtoken ization Ġfor ĠIN 55 50 .*”

Byte-pair encoding (BPE; Gage, 1994)

- Originally an algorithm for text compression
- Now it's the tokenization technique behind most language models, including GPTs
- First we have to *learn* a subword vocabulary (using corpus statistics), then we can tokenize

Byte-pair encoding (BPE; Gage, 1994) – training

Text corpus: *"AABCCAABAABCC"*

- s_0: *"A"*
- s_1: *"B"*
- s_2: *"C"*

Byte-pair encoding (BPE; Gage, 1994) – training

Text corpus: 0 0 1 2 2 0 0 1 0 0 1 2 2

- s_0: *"A"*
- s_1: *"B"*
- s_2: *"C"*

Byte-pair encoding (BPE; Gage, 1994) – training

Text corpus: 0 0-1 2 2 0 0-1 0 0-1 2 2

- s_0: *"A"*
- s_1: *"B"*
- s_2: *"C"*
- s_3: $s_0 + s_1$ = *"AB"*

Byte-pair encoding (BPE; Gage, 1994) – training

Text corpus: 0 3 2 2 0 3 0 3 2 2

- s_0: *"A"*
- s_1: *"B"*
- s_2: *"C"*
- s_3: $s_0 + s_1$ = *"AB"*

Byte-pair encoding (BPE; Gage, 1994) – training

Text corpus: 0-3 2 2 0-3 0-3 2 2

- s_0: *"A"*
- s_1: *"B"*
- s_2: *"C"*
- s_3: $s_0 + s_1$ = *"AB"*
- s_4: $s_0 + s_3$ = *"AAB"*

Byte-pair encoding (BPE; Gage, 1994) – training

Text corpus: 4 2 2 4 4 2 2

- s_0: *"A"*
- s_1: *"B"*
- s_2: *"C"*
- s_3: $s_0 + s_1$ = *"AB"*
- s_4: $s_0 + s_3$ = *"AAB"*

Byte-pair encoding (BPE; Gage, 1994) – training

Text corpus: 4 2-2 4 4 2-2

- s_0: *"A"*
- s_1: *"B"*
- s_2: *"C"*
- s_3: $s_0 + s_1$ = *"AB"*
- s_4: $s_0 + s_3$ = *"AAB"*
- s_5: $s_2 + s_2$ = *"CC"*

Byte-pair encoding (BPE; Gage, 1994) – training

Text corpus: 4 5 4 4 5

- s_0: *"A"*
- s_1: *"B"*
- s_2: *"C"*
- s_3: $s_0 + s_1$ = *"AB"*
- s_4: $s_0 + s_3$ = *"AAB"*
- s_5: $s_2 + s_2$ = *"CC"*

Byte-pair encoding (BPE; Gage, 1994) – training

Text corpus: 4-5 4 4-5

- s_0: *"A"*
- s_1: *"B"*
- s_2: *"C"*
- s_3: $s_0 + s_1$ = *"AB"*
- s_4: $s_0 + s_3$ = *"AAB"*
- s_5: $s_2 + s_2$ = *"CC"*
- s_6: $s_4 + s_5$ = *"AABCC"*

Byte-pair encoding (BPE; Gage, 1994) – training

Text corpus: 6 4 6

- s_0: *"A"*
- s_1: *"B"*
- s_2: *"C"*
- s_3: $s_0 + s_1$ = *"AB"*
- s_4: $s_0 + s_3$ = *"AAB"*
- s_5: $s_2 + s_2$ = *"CC"*
- s_6: $s_4 + s_5$ = *"AABCC"*

Byte-pair encoding (BPE; Gage, 1994) – tokenization

Raw text: *"CCAABCC"*

Vocabulary / merge list

- s_0: *"A"*
- s_1: *"B"*
- s_2: *"C"*
- s_3: $s_0 + s_1$ = *"AB"*
- s_4: $s_0 + s_3$ = *"AAB"*
- s_5: $s_2 + s_2$ = *"CC"*
- s_6: $s_4 + s_5$ = *"AABCC"*

Byte-pair encoding (BPE; Gage, 1994) – tokenization

Raw text: *"CCAABCC"*

- 2 2 0 0 1 2 2

Vocabulary / merge list

- s_0: *"A"*
- s_1: *"B"*
- s_2: *"C"*
- s_3: $s_0 + s_1$ = *"AB"*
- s_4: $s_0 + s_3$ = *"AAB"*
- s_5: $s_2 + s_2$ = *"CC"*
- s_6: $s_4 + s_5$ = *"AABCC"*

Byte-pair encoding (BPE; Gage, 1994) – tokenization

Raw text: *"CCAABCC"*

- 2 2 0 0 1 2 2
- 2 2 0 3 2 2

Vocabulary / merge list

- s_0: *"A"*
- s_1: *"B"*
- s_2: *"C"*
- s_3: $s_0 + s_1$ = *"AB"*
- s_4: $s_0 + s_3$ = *"AAB"*
- s_5: $s_2 + s_2$ = *"CC"*
- s_6: $s_4 + s_5$ = *"AABCC"*

Byte-pair encoding (BPE; Gage, 1994) – tokenization

Raw text: *"CCAABCC"*

- 2 2 0 0 1 2 2
- 2 2 0 3 2 2
- 2 2 4 2 2

Vocabulary / merge list

- s_0: *"A"*
- s_1: *"B"*
- s_2: *"C"*
- s_3: $s_0 + s_1$ = *"AB"*
- s_4: $s_0 + s_3$ = *"AAB"*
- s_5: $s_2 + s_2$ = *"CC"*
- s_6: $s_4 + s_5$ = *"AABCC"*

Byte-pair encoding (BPE; Gage, 1994) – tokenization

Raw text: *"CCAABCC"*

- 2 2 0 0 1 2 2
- 2 2 0 3 2 2
- 2 2 4 2 2
- 5 4 5

Vocabulary / merge list

- s_0: *"A"*
- s_1: *"B"*
- s_2: *"C"*
- s_3: $s_0 + s_1$ = *"AB"*
- s_4: $s_0 + s_3$ = *"AAB"*
- s_5: $s_2 + s_2$ = *"CC"*
- s_6: $s_4 + s_5$ = *"AABCC"*

Byte-pair encoding (BPE; Gage, 1994) – tokenization

Raw text: *"CCAABCC"*

- 2 2 0 0 1 2 2
- 2 2 0 3 2 2
- 2 2 4 2 2
- 5 4 5
- 5 6

→ 2 tokens: *"CC"* + *"AABCC"*

Vocabulary / merge list

- s_0: *"A"*
- s_1: *"B"*
- s_2: *"C"*
- s_3: $s_0 + s_1$ = *"AB"*
- s_4: $s_0 + s_3$ = *"AAB"*
- s_5: $s_2 + s_2$ = *"CC"*
- s_6: $s_4 + s_5$ = *"AABCC"*

Practical considerations

- So-called "pre-tokenization" step (actually our good old word tokenization) is performed first, usually
- We need a mechanism to tell apart splits between words and splits inside words
- Letters are not a good atomic unit (UTF-8 can encode 1 112 064 valid characters) – recent systems use UTF-8 bytes as the atomic unit
- Improved training algorithms: WordPiece (Wu et al., 2016), Unigram tokenizer (Kudo, 2018) or Morfessor systems (Creutz and Lagus, 2007)

Subword tokenization

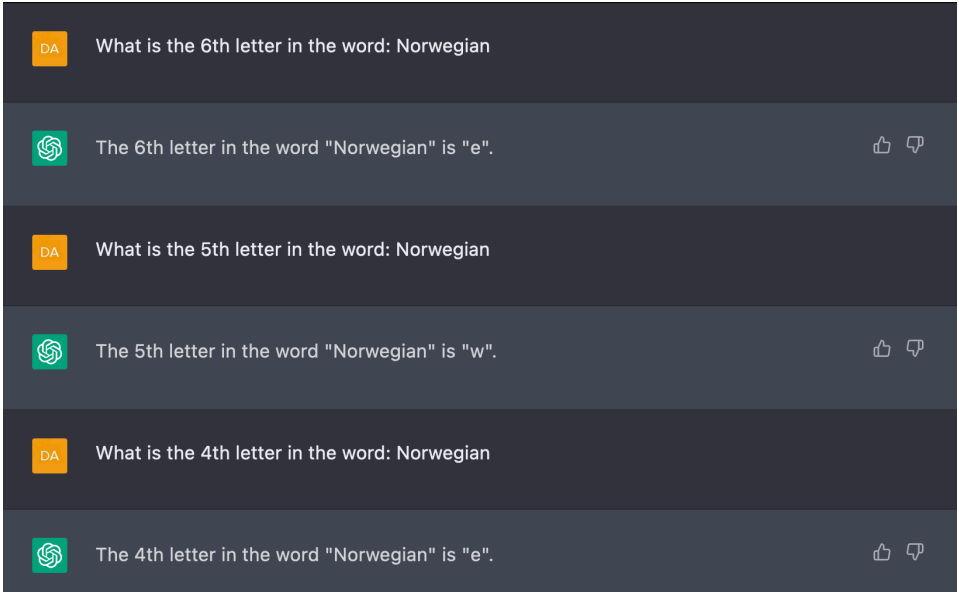


Table of contents

1. Introduction

2. Subword tokenization

3. References

References I

Creutz, M. and Lagus, K. (2007). Unsupervised models for morpheme segmentation and morphology learning. *ACM Trans. Speech Lang. Process.*, 4(1).

Gage, P. (1994). A new algorithm for data compression. *C Users J.*, 12(2):23–38.

Kudo, T. (2018). Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, Melbourne, Australia. Association for Computational Linguistics.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, ., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M., and Dean, J. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation.