BERT

Masked Language modelling and Pre-training

Ryan Chan

07 & 24 July 2023

Outline

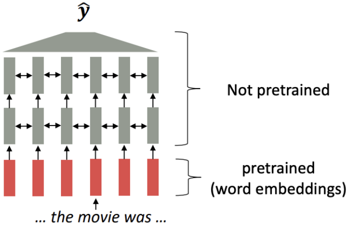
What is pretraining?

Model pretraining involves training a neural network on a large corpus of text to learn language representations. To perform specific downstream tasks, we use the results from pretraining to benefit from the acquired language understanding and improve performance on those tasks

ChatGPT, Accessed 20/07/23

Feature-based approaches

- Before 2017, the dominant paradigm was:
  1. Start with pretrained word embeddings (e.g. via `word2vec` or `GloVe`)
     - No context: "after stealing money from the bank vault, the bank robber was seen fishing on the Mississippi river bank"
     - "Bank" has the same vector representation no matter how it was used in context
  2. Learn how to incorporate context in an LSTM or Transformer while training on a task
- Using pretrained word embeddings worked much better than using embeddings learnt from scratch [Turian et al., 2010; Devlin et al., 2019]
- Key idea: include pretrained representations as additional features in task-specific architectures

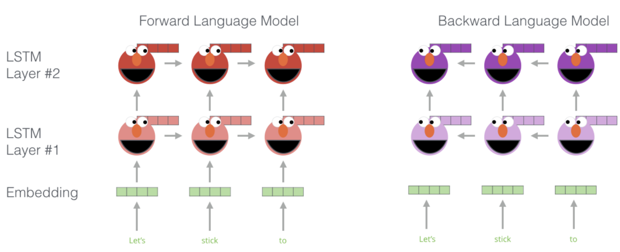


[1]

[1] Manning et al. [2017, Lecture 10]

Contextualised word-embeddings

- ELMo [Peters et al., 2018] generalised traditional word embeddings by extracting *context-sensitive* features:
  - Train a forward *and* backwards language model on a large number of text
  - Language modelling type tasks are useful as we can obtain vasts amount of text data that a model can learn from without needing labels



[2]

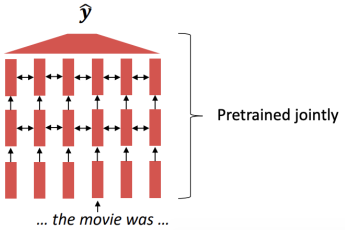
[2] Jay [2018]

Problems with feature-based approaches

- The training data for our downstream task must be sufficient to teach all contextual aspects of language
- Most of the parameters in our network are randomly initialised
- Although ELMo had some concept of bidirectionality, using them was still a feature-based approach to NLP tasks, and also not deeply bidirectional:
  - "ELMo used a shallow concatenation of independently trained left-to-right and right-to-left language models" [Devlin et al., 2019]

Fine-tuning approaches

- In "modern" NLP: All, or almost all, parameters in a network are initialised via pretraining
  - As before, pretraining methods (the tasks that the models are trained on) hide parts of the input from the model, and train the model to reconstruct those parts
- Key idea: introduce only a minimal amount of task-specific parameters, and train the model to perform downstream tasks by simply fine-tuning *all* pretrained parameters



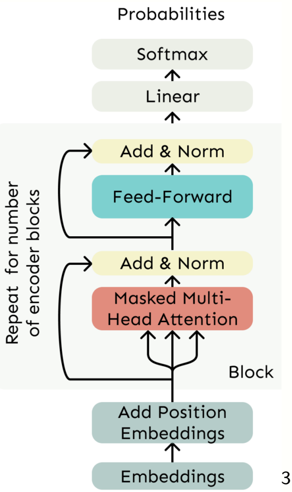
[2]Manning et al. [2017, Lecture 10]

Pretraining transformers

- The transformer encoder-decoder architecture [Vaswani et al., 2017] offered a massive improvement over the previous state-of-the-art LSTM architecture for machine translation
  - Transformers were shown to deal with long-term dependencies much better than LSTMs
  - The encoder-decoder structure of the transformer was perfect for machine translation:
    - The encoder would process the sentence in the source language (bidirectionally)
    - The decoder would process the sentence in the target language
    - We train the entire network end-to-end on the language modelling task on the output target sentence
- Question: how can we pretrain encoder-only and decoder-only models?

Pretraining decoders (GPT)

- OpenAI GPT [Radford et al., 2018] pretrained a decoder model on the standard causal language modelling task: given some tokens, can we predict the next token?
- Radford et al. [2018] obtained (previously) state-of-the-art results on many sentence-level and token-level tasks by pretraining and fine-tuning a unidirectional, left-to-right archiecture:
  - Every token can only attend to previous tokens in the self-attention layers
- Decoders are nice to generate from, but cannot condition on future words

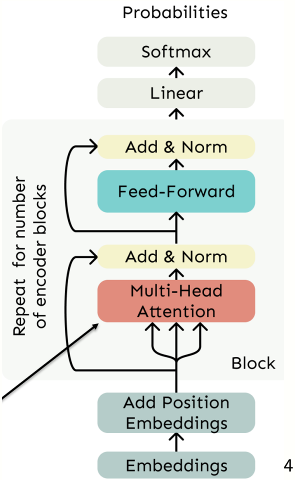


[3]

[3] [Manning et al., 2017, Lecture 8]

BERT: Bidirectional Encoder Representations from Transformers

- Devlin et al. [2019] argue that pretraining decoders is sub-optimal: for many sentence-level taks, it is crucial to incorporate context from both directions
- Encoders allow you to build representations using future words
    - Using the transformer architecture will also improve upon the “shallow bidirectional” ELMo approach
- Question: how do we pretrain transformer encoder?

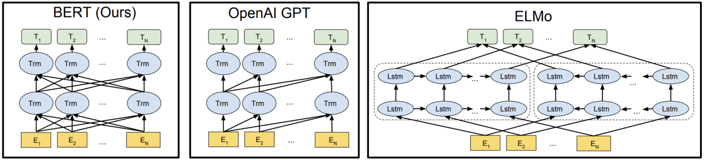


[4]

[4] [Manning et al., 2017, Lecture 8]

BERT

- There are two steps in the BERT framework:
  1. Pretraining
     - Transformer *encoder* model is trained on unlabelled data
  2. Fine-tuning
     - The BERT model is initialised with the pretrained parameters, and *all* of the parameters are fine-tuned using labelled data from some downstream task
- We still have different (fine-tuned) models for different downstream tasks, but they all share the core BERT architecture and initialised with same pretrained parameters



[5]

[5] Devlin et al. [2019, Figure 3]

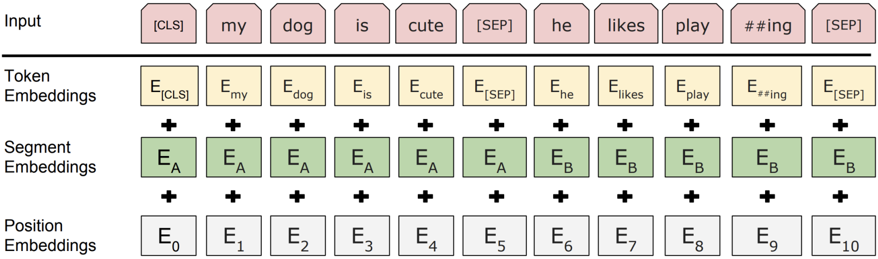
Pretraining BERT

- Unlike ELMo [Peters et al., 2018] and OpenAI GPT [Radford et al., 2018], BERT [Devlin et al., 2019] does not use traditional left-to-right or right-to-left language models to pretrain, but use two unsupervised learning tasks:
    1. Masked Language Modelling (MLM)
    2. Next Sentence Prediction (NSP)

Input representations in BERT

- We want to build an architecture which can be easily fine-tuned to a variety of downstream tasks
  - to excel at both token-level and sentence-level tasks
  - to work with single sentences and pairs of sentences
- To enable this, BERT introduces some new special tokens:
  - The first token of every sequence is always the special classification token: `[CLS]`
    - The final hidden state corresponding to this token is always used as the sequence representation for classification tasks
  - Pairs of sentences are differentiated in two ways:
    1. Pairs of sentences are separated with a special separation token: `[SEP]`
    2. Introduce learned embeddings for indicating whether or not a token belongs to sentence `A` or sentence `B`

Input representations in BERT



[6]

[6] Devlin et al. [2019, Figure 2]

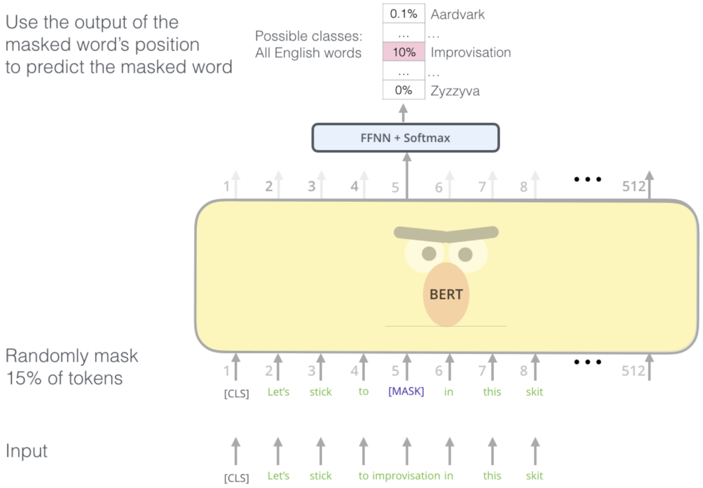
Masked Language Modelling (MLM)

- For encoders, we cannot use the standard causal language modelling task
  - bidirectional conditioning in the encoder would allow each word to see all other tokens in the sequence
  - the model could trivially predict the next target word in a multi-layered network
- Instead, we can pretrain transformer encoders using the masked language modelling task (also known as the Cloze task [Taylor, 1953])
  1. Simply mask a percentage of tokens at random, and predict those masked tokens
  2. The final hidden vectors corresponding to the mask tokens are fed into an output softmax over the vocabulary (like we do in standard language modelling)
- "I went to the store" $\rightarrow$ "I [MASK] to the [MASK]"

MLM in BERT

- *Do not* replace all of these with the special [MASK] token
    - Doing this alone creates a problem in that we create a mismatch between the pre-training and fine-tuning data, as [MASK] does not appear in fine-tuning
- In BERT:
    - 15% of tokens are sampled at random for prediction
        - 80% of these tokens are replaced with [MASK]
        - 10% are replaced with a random token in the vocabulary
        - 10% are unchanged

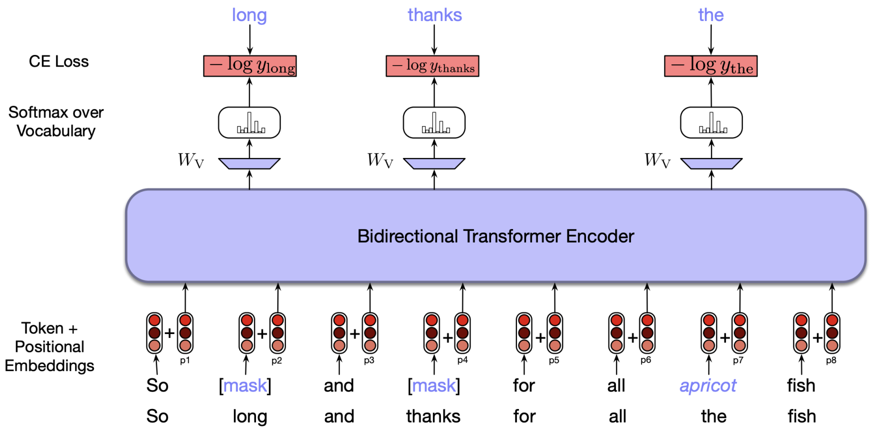
MLM in BERT



[7]

[7] Jay [2018]

MLM in BERT

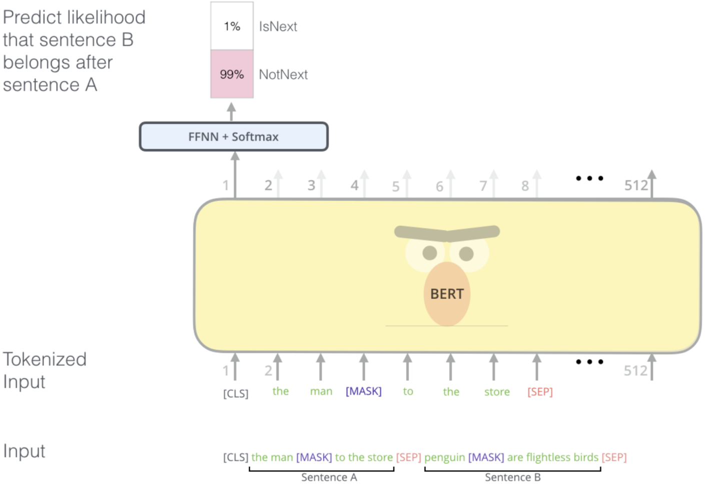


[8] Jurafsky and Martin [2019, Chapter 11]

Next Sentence Prediction (NSP)

- Many downstream tasks, like question answering, are based on understanding the relationship between two sentences
  - Devlin et al. [2019] argued this is not directly captured by language modelling
- To train BERT to understand sentence relationships, it is also pretrained to the Next Sentence Prediction (NSP) task:
  - When choosing sentences `A` and `B` to construct the pre-training data:
    - 50% of the time, `B` actually is the next sentence that follows `A`: given `IsNext` label
    - 50% of the time, `B` is a random sentence from the corpus: given `NotNext` label
- The final hidden state representation for the `[CLS]` token passed into a two-class output softmax

Next Sentence Prediction (NSP)



[9]

[9] Jay [2018]

BERT architecture and pretraining setup

- Training data ($\approx$ 3.3 billion words)
  - 800 million word corpus of book texts from `BookCorpus` - no longer used anymore for intellectual property reasons
  - 2.5 billion word corpus from English Wikipedia
- Architecture largely follows the same as the multi-layer bidirectional Transformer Encoder described in Vaswani et al. [2017]
  - Let L be number of layers/transformer blocks, H be hidden dimension size, A number of attention heads:
    - $\texttt{BERT}_{\texttt{BASE}}$: $L = 12, H = 768, A = 12$ ($\approx$ 110M parameters)
    - $\texttt{BERT}_{\texttt{LARGE}}$: $L = 24, H = 1024, A = 16$ ($\approx$ 340M parameters)
  - Context length of 512 tokens
- Llama 2 is trained on 2 trillion tokens and number model parameters range from 7B to 70B with 4096 token context window

Contextual word-embeddings from BERT

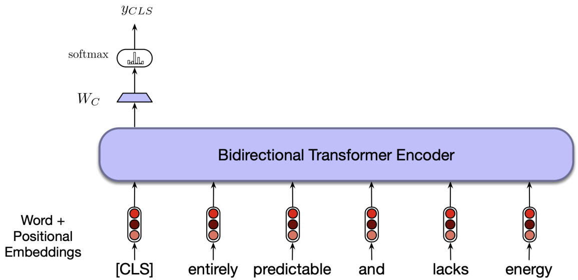
- Given a pretrained language model and an input, we can think of the output of BERT as constituting contextual embeddings for each token in the input
  - These can be used as a contextual representation of the *meaning* of a word
  - Can be used as features in some downstream task, e.g. named entity recognition, parts-of-speech tagging
- For a given sequence of input tokens x_i, we can use the output vector $\boldsymbol{y}_i$ from the final layer as the representation of the meaning of token x_i *within* the context of the sentence $x_1, \ldots, x_n$
- Note: there are several ways to use the hidden layers of the transformer:
  - Take outputs of the second-to-last layer
  - Concatenate the outputs of the last four layers
  - Take the sum of several layers
  - ...

Fine-tuning BERT

- The power of pretrained models lies in their ability to extract generalisations from large amounts of text
  - To make pretrained models practical, we need to create interfaces from these general models to more specific applications - this is fine-tuning
  - Fine-tune facilitates creation of applications to be placed on-top-of pretrained models through addition of a small set of parameters
- Pretraining task $\rightarrow$ training on a large corpus of unlabelled data ("semi-supervised")
- Fine-tuning $\rightarrow$ training on labelled data from a specific application to train these additional application-specific parameters
  - transfer learning in machine learning: the method of acquiring knowledge from one task or domain, and then applying it (or *transferring* it) to solve a new task

Fine-tuning BERT: sequence classification

- In sequence classification, we aim to classify the entire sequence (rather than the individual elements/tokens):
  - e.g. sentiment analysis, spam detection, document-level topic classification
- For BERT, the final hidden state for the special [CLS] token is used as the sequence representation for classification tasks

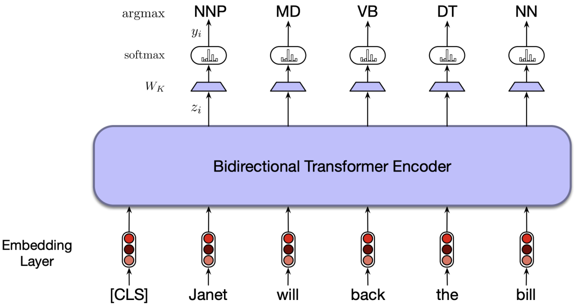


10

[10] Jurafsky and Martin [2019, Chapter 11]

Fine-tuning BERT: sequence labelling

- In sequence labelling, we aim to assign a label to each element in the sequence:
  - e.g. part-of-speech tagging, named entity recognition
- We can simply add a linear layer with softmax to predict the output classes on each token of interest - just like MLM



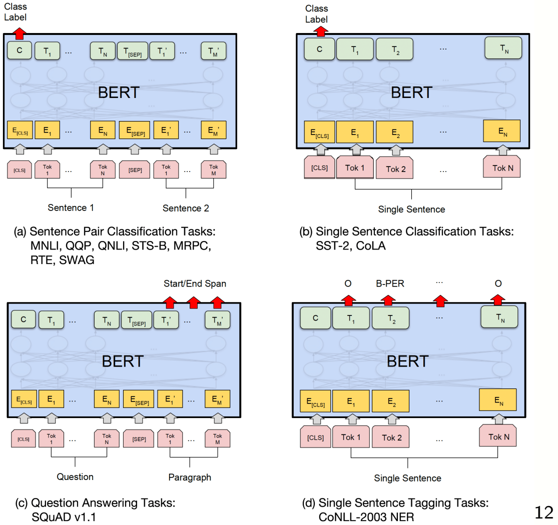
11

[11] Jurafsky and Martin [2019, Chapter 11]

Fine-tuning BERT: pairwise sentence classification

- Many problems in NLP involve the classification of *pairs* of sentences
  - recognising textual entailment (natural language inference (NLI)), paraphrase detection, etc.
- Fine-tuning a pair-wise sequence classification task is just like how we pretrain with the next-sentence prediction (NSP) objective:
  - Split pairs of sentences using the `[SEP]` token and the learned embeddings for denoting sentences `A` and `B`
  - Peform classification on the pair-wise sentence task using the `[CLS]` token (i.e. multiply by classification weights and apply softmax to generate label predictions)

Fine-tuning BERT



12

[12] Devlin et al. [2019, Figure 4]

Summary of results in BERT

- Fine-tuning BERT led to new state-of-the-art results on a broad range of tasks
- Evaluated on the General Language Understanding Evaluation (GLUE) benchmark [Wang et al., 2018], a collection of diverse natural language understanding tasks

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	79.8	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
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

[13]

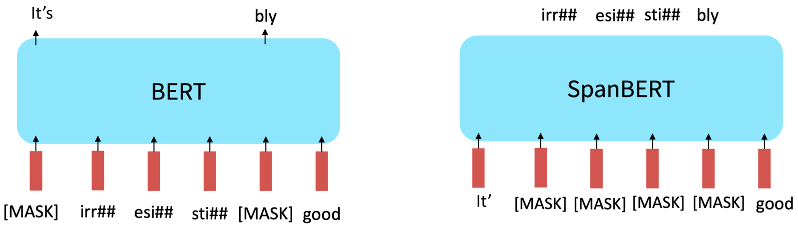
[13] [Devlin et al., 2019, Table 1]

Limitations of pretrained encoders

- Why not use pretrained encoders for everything?
- If task involves generating sequences, must have some kind of decoder architecture
    - BERT and other pretrained encoders don't naturally lead to nice autoregressive generation methods
- But they should be used in settings where you want to build good representations (encode) about some input text

Some extensions of BERT

- There are many variants of BERT now, like RoBERTa [Zhuang et al., 2021], SpanBERT [Joshi et al., 2020], DistilBERT [Sanh et al., 2019], ALBERT [Lan et al., 2019], ...
  - RoBERTa: more-or-less just BERT, but train it for longer and remove NSP
    - Showed that you can actually get really much better results by simply just training for longer without changing anything about the underlying model
  - SpanBERT: masking contiguous spans of words
    - Makes the pretraining task harder, and maybe more useful pretraining task
  - DistilBERT, ALBERT: smaller, cheaper, lighter alternatives to BERT



[14]

[14] [Manning et al., 2017, Lecture 9]

References

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Jay, A. (2018). The Illustrated BERT.

Joshi, M., Chen, D., Liu, Y., Weld, D. S., Zettlemoyer, L., and Levy, O. (2020). SpanBERT: Improving Pre-training by Representing and Predicting Spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

Jurafsky, D. and Martin, J. H. (2019). Speech and language processing (3rd (draft) ed.).

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. (2019). Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

Manning, C., Socher, R., Fang, G. G., and Mundra, R. (2017). CS224n: Natural Language Processing with Deep Learning.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al. (2018). Improving Language Understanding by Generative Pre-Training. Technical report.

Sanh, V., Debut, L., Chaumond, J., and Wolf, T. (2019). DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Taylor, W. L. (1953). Cloze procedure: A new tool for measuring readability. *Journalism Bulletin*, 30(4):415–433.

Turian, J., Ratinov, L., and Bengio, Y. (2010). Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual meeting of the Association for Computational Linguistics*, pages 384–394.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention Is All You Need. *Advances in Neural Information Processing Systems*, 30.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. (2018). GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Zhuang, L., Wayne, L., Ya, S., and Jun, Z. (2021). A Robustly Optimized BERT Pre-training Approach with Post-training. In *Proceedings of the 20th Chinese National Conference on Computational Linguistics*, pages 1218–1227, Huhhot, China. Chinese Information Processing Society of China.