Training BERT from scratch (a brief tutorial)

Antti Virtanen, Sampo Pyysalo, Filip Ginter

Turku NLP group, University of Turku, Finland
www.turkunlp.org

TurkuNLP

UNIVERSITY OF TURKU

Why make language-specific BERT models?

- NLPL DL Workshop paper at NoDaLiDa’19

Is Multilingual BERT Fluent in Language Generation?

Samuel Rönnqvist* **Jenna Kanerva*** **Tapio Salakoski** **Filip Ginter***
TurkuNLP
Department of Future Technologies
University of Turku, Finland
{saanro,jmnybl,sala,figint}@utu.fi

Abstract

The multilingual BERT model is trained

gual BERT model as the availability of pre-trained BERT models for other languages is extremely scarce. For the vast majority of languages, the only option is the multilingual BERT model

TurkuNLP

UNIVERSITY OF TURKU

Is multilingual BERT good enough?

- In the workshop paper, we ran several tests on the multilingual BERT and compared it, where possible, to the monolingual variants
  - Diagnostic classifier on syntax inspired by Lin et al. 2019. “Open sesame: Getting inside bert’s linguistic knowledge”. Is the given token an auxiliary of the main verb in the sentence? Y/N
  - Cloze task: 15% of words (not sub-words) masked, model fills in
  - Given a left and right sentence, generate a sentence in between (sentence-level cloze) Generation done as in Wang and Cho. 2019. “BERT has a mouth, and it must speak: BERT as a Markov random field language model”

Highlights from the evaluation 1/3: diagnostic class.

Language	BERT	Test acc.	Baseline
English	mono	86.03	54.93
	multi	87.82	54.44
German	mono	97.27	69.61
	multi	95.29	69.19
Danish	multi	89.96	53.25
Finnish	multi	93.20	50.54
Nor. (Bokmål)	multi	93.67	56.19
Nor. (Nynorsk)	multi	94.44	53.18
Swedish	multi	93.00	62.09

Table 1: Diagnostic classifier results. Auxiliary classification task accuracies and majority class baselines for all languages.

Highlights from the evaluation 2/3: Cloze task

	Mono	Multi
English	**45.92**	33.94
German	**43.93**	28.10
Swedish		22.30
Finnish		14.56
Danish		25.07
Norwegian (Bokmål)		25.21
Norwegian (Nynorsk)		22.28

Table 2: Results for the cloze test in terms of sub-word predictions accuracy.

		match	mismatch	copy	gibb
Eng	mono	**88%**	9%	1%	1%
	multi	**72%**	15%	8%	6%
Ger	mono	**82%**	12%	1%	5%
	multi	**69%**	15%	6%	10%
Fin	multi	**42%**	15%	3%	39%
Swe	multi	**56%**	19%	2%	23%

Table 3: Manual evaluation of words generated in the cloze test.

TurkuNLP

UNIVERSITY OF TURKU

Highlights from the evaluation 3/3: Generation

		on-top	off-top	copy	gibb
Eng	mono	**50%**	21%	5%	24%
	multi	7%	2%	38%	**53%**
Ger	mono	**67%**	28%	3%	2%
	multi	17%	13%	**48%**	22%
Fin	multi	19%	2%	37%	**43%**
Swe	multi	10%	5%	**47%**	37%

Table 4: Manual evaluation of generated text from the mono- and multilingual models. The categories are, in order, on-topic original text, off-topic original text, copy of the context, and gibberish. N is 55–60 for all tests.

Evaluation: conclusions

- Multilingual BERT lags behind the monolingual models
- The gap opens notably as task complexity increases
- Why?
  - Vocabulary capacity: ~100K items is not enough for ~100 languages
  - Model capacity: ~110M parameters may not be enough for ~100 languages
  - Size of source data: Wikipedias are quite small for a lot of languages
  - Considerably smaller training effort per-language (in terms of updates)
- Practical conclusion: must train own model!

Vocabulary effect example:
FinBERT: Suomessa vaihtuu kesän aikana sekä pääministeri että valtiovarain ##ministeri .
M-BERT: Suomessa vai ##htuu kes ##än aikana sekä p ##ää ##minister ##i että valt ##io ##vara ##in ##minister ##i .

FinBERT

- Preprint on Arxiv

Multilingual is not enough: BERT for Finnish

Antti Virtanen[1] Jenna Kanerva Rami Ilo[2] Jouni Luoma[3] Juhani Luotolahti
Tapio Salakoski Filip Ginter Sampo Pyysalo
Turku NLP group, University of Turku
[1]sajvir@utu.fi, [2]rajuil@utu.fi, [3]jouni.a.luoma@utu.fi, first.last@utu.fi

TurkuNLP

UNIVERSITY OF TURKU

FinBERT in nutshell: training data

- Three text sources
- Dedup + heavy filtering for quality
- 3.3B tokens left

	Docs	Sents	Tokens	Chars
News	4M	68M	0.9B	6B
Discussion	83M	351M	4.5B	28B
Crawl	11M	591M	8.1B	55B
Total	98M	1 010M	13.5B	89B

Table 1: Pretraining text source statistics. Tokens are counted using BERT basic tokenization.

	Docs	Sents	Tokens	Chars
News	3M	36M	0.5B	4B
Discussion	15M	118M	1.7B	12B
Crawl	3M	79M	1.1B	8B
Total	21M	234M	3.3B	24B

Table 2: Pretraining text statistics after cleanup and filtering

FinBERT results: POS

	TDT		FTB		PUD	
FinBERT cased	**98.23**	(0.04)	**98.39**	(0.03)	**98.08**	(0.04)
FinBERT uncased	98.12	(0.03)	98.28	(0.07)	97.94	(0.03)
M-BERT cased	96.97	(0.06)	95.87	(0.09)	97.58	(0.03)
M-BERT uncased	96.59	(0.05)	96.00	(0.07)	97.48	(0.03)
(Che et al., 2018)	97.30	—	96.70	—	97.60	—
(Lim et al., 2018)	97.12	—	96.20	—	97.65	—

Table 6: Results for POS tagging (standard deviation in parentheses)

FinBERT results: NER

	Prec.		Rec.		F1	
FinBERT cased	**91.30**	(0.12)	**93.52**	(0.10)	**92.40**	(0.09)
FinBERT uncased	90.37	(0.35)	92.67	(0.19)	91.50	(0.24)
M-BERT cased	89.35	(0.21)	91.25	(0.17)	90.29	(0.14)
M-BERT uncased	88.07	(0.25)	90.07	(0.22)	89.06	(0.21)
Rule-based → FiNER-tagger	90.41	—	83.51	—	86.82	—
ML baseline → (Güngör et al., 2018)	83.59	—	85.62	—	84.59	—

Table 8: NER results for in-domain test set (standard deviation in parentheses)

	Prec.		Rec.		F1	
FinBERT cased	80.61	(0.61)	**82.35**	(0.33)	**81.47**	(0.46)
FinBERT uncased	80.74	(0.31)	79.38	(0.68)	80.05	(0.42)
M-BERT cased	75.60	(0.49)	76.71	(0.61)	76.15	(0.50)
M-BERT uncased	75.73	(0.73)	71.93	(1.01)	73.78	(0.81)
FiNER-tagger	**88.66**	—	72.74	—	79.91	—
(Güngör et al., 2018)	67.46	—	55.07	—	60.64	—

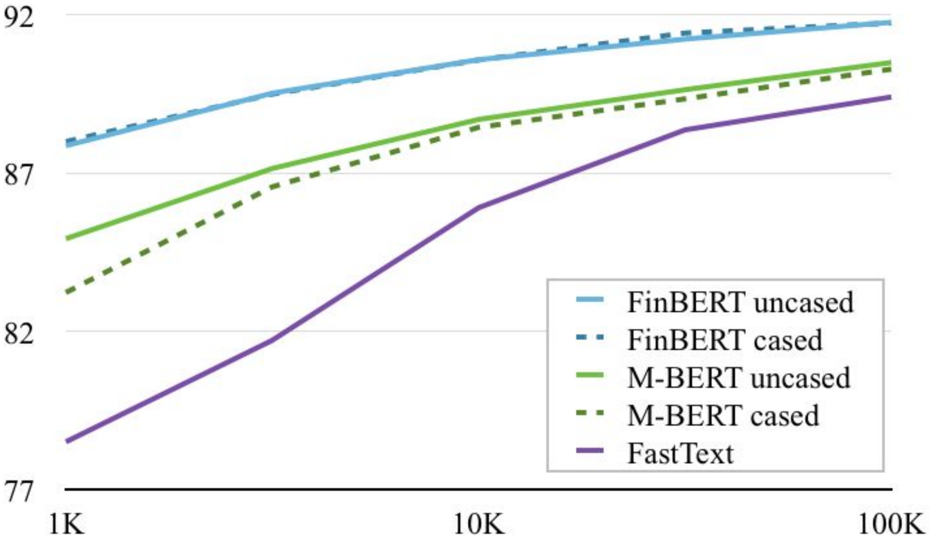
Table 9: NER results for out of domain test set (standard deviation in parentheses)

TurkuNLP

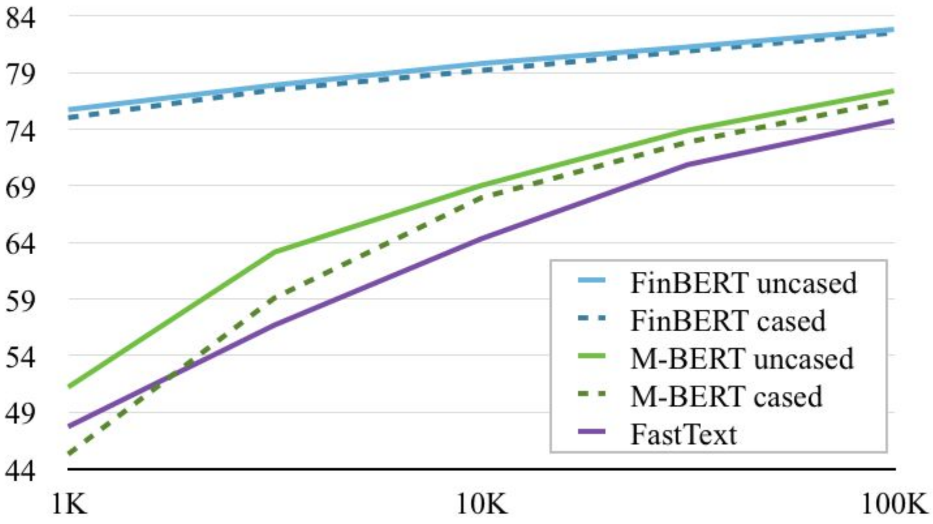
UNIVERSITY OF TURKU

FinBERT results: Document classification

News



Online discussion



TurkuNLP

UNIVERSITY OF TURKU

FinBERT results: UD parsing

	TDT		FTB		PUD	
Model	p.seg.	g.seg	p.seg.	g.seg.	p.seg	g.seg.
FinBERT cased	**91.93**	**93.56**	**92.16**	**93.95**	**92.54**	**93.10**
FinBERT uncased	91.73	93.42	91.92	93.63	92.32	92.86
M-BERT cased	86.32	87.99	85.52	87.46	89.18	89.75
M-BERT uncased	86.74	88.61	86.03	87.98	89.52	89.95
(Che et al., 2018)	88.73	—	88.53	—	90.23	—
(Kulmizev et al., 2019)	—	87.0*	—	—	—	—

Table 10: Labeled attachment score (LAS) parsing results for for predicted (p.seg) and gold (g.seg) segmentation. *Best performing combination in the TDT treebank (ELMo + transition-based parser).

UDify parser: Kondratyuk and Straka. 2019. “75 Languages, 1 Model: Parsing Universal Dependencies Universally” (Arxiv)

FinBERT results: Conclusions

Evaluation on downstream tasks

- Multilingual BERT roughly comparable with prior state of the art
  - Better in some tasks, worse in others
- Monolingual FinBERT gives top performance on all tested tasks
  - Often by a substantial margin

Conclusion:

Don’t rely on the multilingual BERT, train your own!

...you will be in good company

Latest list of BERT models is on twitter (where else)
https://twitter.com/seb_ruder/status/1221851361811128321

German BERT and dbmdz BERT (**German**), CamemBERT and FlauBERT (**French**), ALBERTo and GilBERTo (**Italian**), RobBert and BERTje (**Dutch**), RuBERT (**Russian**), BETO (**Spanish**), **Portugese** BERT, **Danish** BERT, FinBERT (**Finnish** but also Financial), **Chinese** BERT, **Japanese** BERT, **Swedish** BERT and of course **English** BERT

...did we miss any?

Text preprocessing for BERT

BERT input is WordPiece-tokenized and training involves next sentence prediction[1], where document boundaries are used to select negative examples

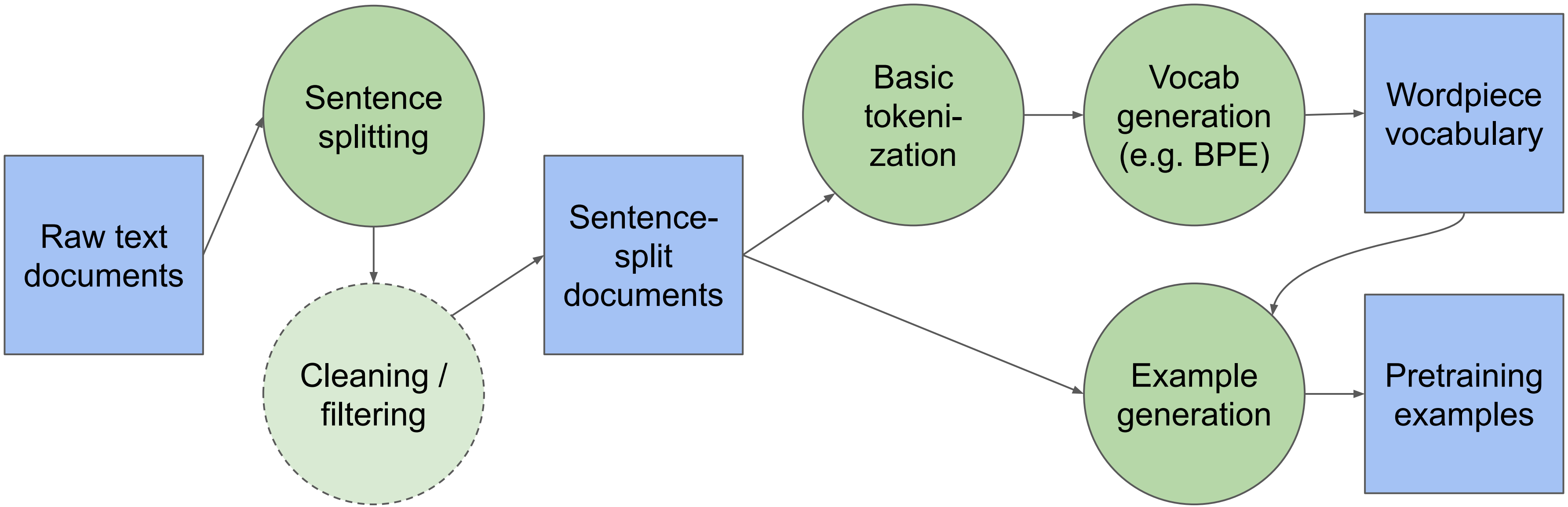
- Need a WordPiece vocabulary
- Need document boundaries in input
- Need sentence segmentation

For efficiency, the original BERT implementation expects its input as TFRecords

- Need to generate TFRecords from corpus texts

[1] The value of next sentence prediction is disputed; included here as part of original BERT

A text preprocessing pipeline for BERT training



TurkuNLP

UNIVERSITY OF TURKU

Text cleaning / filtering

Many large-scale text sources (esp. web crawls) are noisy

- Markup, markdown, tags, metadata
- Mix of different languages (incl. artificial)
- Spam, machine-generated and machine translated text
- High redundancy (incl. headers, menus, legalese, etc.)
- Text encoding errors
- …

Use known-good sources (e.g. Wiki+BookCorpus) and/or include cleaning and filtering in your pipeline

Text cleaning / filtering

Monolingual resources are not always monolingual ...

From **Finnish Wikipedia** (“*Lunch Money*”)

"You have sinned. You must be punished"
"Twinkle, twinkle, little star; how you wonder who you are."
"Trust me, I'm a dentist"
"Hi!"
"Kick You Once. Kick You Twice. Next time I WON't be so nice"
"Look at me when I'm hitting you."

From **BookCorpus** (English): “*1000 Japanese Flash Cards: For Smart Phones and E-Readers*”

Ohayou gozaimasu. おはようございます。
Konban wa. こんばんは。
Oyasuminasai. おやすみなさい。
Sayounara. さようなら。
Ja ne. じゃね。
Mata ne. またね。
Mata ashita. またあした。
Ki o tsukete. きをつけて。
Moshi moshi もしもし。

Text cleaning / filtering

Possible text cleaning / filtering steps

- Deduplication (e.g. http://corpus.tools/wiki/Onion)
- Language detection (e.g. https://pypi.org/project/langdetect/)
- Encoding correction (e.g. https://pypi.org/project/ftfy/)
- Filtering heuristics, e.g.
  - Average word / sentence length (segment e.g. w/http://ufal.mff.cuni.cz/udpipe)
  - Ratio of non-alphanumeric characters, foreign letters, uppercase, etc.
- Perplexity per word vs. known-good text (e.g. Wiki+N-gram LM)
- ...

Pretokenization and vocabulary generation

BERT pretokenization (“basic tokenization”) separates punctuation and in “uncased” mode also lowercases and removes accents (e.g. “äö” -> “ao”)

Vocab generation implementations (e.g. SentencePiece) generally don’t

→ Generating a vocabulary on raw text may generate pieces that BERT cannot use (e.g. uppercase, “ä”, “##?) and may lack common ones (e.g. “?”)

Vocab generation input should match the pretokenization of the tool the vocab will be used with

Practical vocabulary generation example

Using https://github.com/google/sentencepiece (Kudo and Richardson 2018)

```
spm_train
    --input=<TOKENIZED-TEXT>
    --model_prefix=<NAME>
    --model_type=bpe
    --vocab_size=20000
    --character_coverage=0.9999
    --input_sentence_size=1000000
    --shuffle_input_sentence=true
```

NOTES:

- bpe is close to the unavailable algo used with original BERT
- character_coverage <1.0 avoids random rare unicode
- input_sentence_size and shuffle_input_sentence for sampling (faster)

(convert to WP conventions with e.g. https://github.com/spyysalo/sent2wordpiece)

Example generation

BERT pretraining examples are shuffled by interleaving (sequentially read) examples from set of shuffled input TFRecords

→ Don't just make one big TFRecord, shuffling wouldn't work

The original BERT pretraining process uses a sequence length of 128 for the initial 90% of steps and 512 for the last 10%

→ Need to generate two sets of examples (sequence length 128 and 512)

NB: <u>max predictions per sequence</u> should be adjusted for sequence length (e.g. 20 for 128 and 77 for 512)

Practical example generation example

Using https://github.com/google-research/bert

```
python3 bert/create_pretraining_data.py
	--input_file=<SEGMENTED-TEXT>
	--output_file=<NAME>
	--vocab_file=<VOCAB>
	--do_whole_word_mask=true
	--do_lower_case=false
	--dupe_factor=10
	--max_seq_length=128
	--max_predictions_per_seq=20
```

NOTES:

- example for cased, seq 128
- do_whole_word_mask used in most recent BERT models
- do_lower_case=true for uncased also strips accents
- dupe_factor should be adjusted for amount of text

Quickstart

Vocab and example generation takes time we don't have just now

To allow experimenting on actual BERT training, we've prepared vocabularies and TFRecords for you from Wikipedia texts in a few languages:

http://dl.turkunlp.org/nlpl-2020/

(Generating pipeline: https://github.com/spyysalo/wiki-bert-pipeline)

BERT pretraining

To download tfrecords for a language you can do:

wget -nH --recursive --no-parent --cut-dirs=2 http://dl.turkunlp.org/nlpl-2020/tfrecords/<LANG>/seq-128/ --reject "index.html"

Where <LANG> in {da, es, fi, no, sv}

Vocabs in http://dl.turkunlp.org/nlpl-2020/vocabs/ (not needed in pretraining)

Continued:
https://github.com/TurkuNLP/FinBERT/blob/master/nlpl_tutorial/training_bert.md

TurkuNLP

UNIVERSITY OF TURKU