CYUT at the NTCIR-15 FinNum-2 Task: Tokenization and Fine-tuning Techniques for Numeral Attachment in Financial Tweets

Mike Tian-Jian Jiang
Zeals Co, Ltd.
Tokyo, Japan
tmjiang@gmail.com

Yi-Kun Chen
Department of CSIE
Chaoyang University of Technology
Taichung, Taiwan
kun26712930@gmail.com

Shih-Hung Wu†
Department of CSIE
Chaoyang University of Technology
Taichung, Taiwan
shwu@cyut.edu.tw

ABSTRACT

The paper describes our submissions to the NTCIR-15-FinNum-2 shared task in financial tweets analysis. We submitted two runs in the final test. The first run is our baseline system, which is based on the BERT model with our preprocessing strategy. The second run is our fine-tuned system based on the XLM-RoBERTa pretraining model with more tokenization and fine-tuning techniques. The macro-F1 of run 2 is 95.99% on development set, and 71.90% on formal test which ranked second best.

CCS CONCEPTS

• Information systems→Information extraction.

KEYWORDS

Financial social media, Financial tweets, BERT, XLM-RoBERTa.

TEAM NAME

CYUT

1 Introduction

FinNum-2 is a shared task to analyze financial tweets, these tweets are discussing stock prices and the companies [5]. There are many stock names and many numbers in these tweets. The goal of the shared task is to uncover whether names and numbers in a tweet associates or not. In the training data, we can see that there is at least one pair of target numeral and cashtag in a tweet, therefore the problem definition can be a binary classification to tell if the target numeral is relevant to the given cashtag.

Most submissions of NTCIR-14 FinNum-1 [3,4] use word/character embeddings to represent token information of tweets, namely Skip-grams [22], GloVe [17], ELMo [1], and the Bidirectional Encoder Representations from Transformers (BERT) [29,30]. A BERT [7] model pretrained with Microsoft Research Paraphrase Corpus (MRPC) has obtained the best performance [29,30]. Consequently, the result of FinNum-1 inspires us to further explore most recent Transformer [28] models pretrained with different datasets and tokenization schemes. In order to efficiently examine various pretrained models, we also apply several fine-tuning techniques of learning rate and optimizer.

The rest of this paper is organized as follows, in the second section, we will give the detail of our approaches. In the third section, we will show the experiment setting and results, also discussions on the results. In the final section, we will give conclusion and future works.

2 Proposed Approaches

For comparison, we fine-tune two Transformer models, BERT and XLM-RoBERTa [6]. Both pretrained models are the base versions. Please note that the base BERT model we use is lower-cased, but XLM-RoBERTa, one the other hand, always preserves letter cases. For the latter one, we further apply techniques developed by fastai [8,9] for ULMFiT [10], such as discriminative fine-tuning and a variation of One-cycle policy [25,26]. This section will only introduce model specifications and training procedures that are conceptually related to our attempt of the FinNum-2. Please kindly refer to the original papers of those models and techniques for further details.

2.1 Model 1 BERT

In the CYUT-1 run, we build a baseline system based on a pretrained BERT model. Pretraining of a Transformer-based language model typically relies on two objectives: masked language modeling (MLM) and next sentence prediction (NSP). For BERT pretraining, the former requires the model to predict tokens that have been randomly masked in a 15% chance per input sentence, and the latter demands the model to predict whether two randomly concatenated sentences are actually adjacent to each other or not.

Before fine-tuning the BERT model, we apply a preprocessing strategy to the data that normalizes all cashtag instances as one representative tag and all numerals as one designated symbol. The strategy is based on an assumption that the exactly the same cashtags or numerals, along with their attachments, might not appear in the test set, so we treat them as identical ones, and then expect the model to be more focused on learning the patterns of the

†Contact Author

context. After the preprocessing, we tokenize tweets with WordPiece [32], following the instructions of the BERT-based previous works [29,30] of FinNum-1, and fine-tune the model using HuggingFace's Transformers [31].

2.2 Model 2 XLM-RoBERTa

XLM-RoBERTa [6] is a Transformer model trained with the multilingual MLM objective. The model combines and revises techniques of cross-lingual language model (XLM) pretraining schemes [16] and a robustly optimized BERT pretraining approach (RoBERTa) [19]. In terms of optimization, RoBERTa builds on BERT and modifies key hyperparameters such as the MLM objectives, removing the NSP objective and training with much larger mini-batches and learning rates. As for tokenization, it differs from BERT by using a byte-level Byte Pair Encoding (BPE) [24] as a tokenizer, and dynamically changing the masking pattern applied to the training data. XLM-RoBERTa follows most of XLM approaches, except it removes language embeddings for a better code-switching ability. It also differs from RoBERTa by tokenizing with unigram-level sentencepiece [14,15] instead of BPE.

2.3 Tokenization Tricks

To better represent the structure of a financial tweet, we not only utilize XLM-RoBERTa's special tokens, namely the beginning of a sentence (`<s>`), the end of a sentence (`</s>`), and the separator of sentences (`</s> </s>`), but also customize a couple of tokens in the fastai convention of "`xx`" prefix that provides context. For example, consider a tokenized tweet below:

```
<s> _$ xxtag _RAD _about xxnum _9
_million _more _share s _than _the _90
_day _average . … </s>
```

The special tokens `xxnum` and `xxtag` annotate the numeral (`_9` but not `_90`) and the cashtag (`_RAD`) in question, respectively. Combining with the actual subwords of number/cashtag right next to `xxnum`/`xxtag`, the annotated tokens provide certain features of the token sequence.

Although we don't apply the default tokenizer of fastai, it might be worthwhile to explain what it is and why we don't use it. The fastai convention of "`xx`" prefix denotes special context tokens. By default, fastai tokenizes English texts using SpaCy and inserts special tokens before uncapitalized or originally repeated words/characters[1]. For instance, consider the following two tweets from the test set:

```
$TSLA DHL ordered 10 Semis … at any moments
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

$FOXA … bully bully BUY BUY … 20/20 lol
```

If we apply fastai's default tokenization to the first tweet, the thirty-five characters of "`$`" at its tail will become "`xxrep 35 $`" without loss of generality. Similarly, the second tweet's tokenization outcome will have "bully bully BUY BUY" converted into "`xxwrep 2 bully xxwrep 2 xxup buy`" for all-capital and recurring words simultaneously. As lossless as the conversion may be, since pretrained Transformer models are unaware of those special context tokens, we must ask whether they can still help fine-tuning for a specific task or not. In our opinions, if the task were sentiment analysis of tweets, repetitions and capitalization could be important clues. However, even if the digits coming from those special context tokens won't negatively impact the numeral attachment task, it is still hard to imagine that the lengths of word/character duplications can help semantically or syntactically, not to mention that XLM-RoBERTa already preserves letter cases of subword tokens. Based on the above observations, we don't apply them to the FinNum-2 task.

2.4 Fine-tuning Techniques

We adopt recently advanced fine-tuning techniques as much as possible. Some of them are originally designed for AWD-LSTM and QRNN [20,21] by ULMFiT, such that we must assess their usefulness for XLM-RoBERTa. Based on our preliminary tests, discriminative fine-tuning and fastai's version of one-cycle policy work well, but graduate unfreezing produces little effect, which is consistent with the findings of similar studies [11,23]. Techniques other than the above mainly involve choosing the most promising combination of optimization algorithms and loss functions. For the FinNum-2 task in a binary classification setting, we find none of more recent optimizers and loss functions work better than Adam optimizer with class weights. We will list configuration values of finally used techniques in the next section of experiments. The section of related works will briefly describe what optimizers and loss functions we have evaluated.

2.4.1 Discriminative Fine-tuning. As different layers may capture various types of information, we shall fine-tune them to different extents. Instead of using the same learning rate for all layers of the model, discriminative fine-tuning enables us to tune each layer with different learning rates. We use blurr[2] to split the model layers into groups automatically corresponding to architectures. In terms of XLM-RoBERTa, it results in four groups, ranging from the top layer of classifier, the pooling layer, the Transformer layers, to the bottom layer of embeddings. Intuitively, the lower groups may contain more general information while the higher ones contain more specific information. Therefore, we set a base learning rate for the top group and then assign linearly decreased learning rates per lower groups.

2.4.2 One-cycle Policy. A cycle wraps an arbitrary number of epochs for sharing the same policy of hyperparameters, especially for learning rates and momentums. For training a deep neural network with stochastic gradient decent or similar algorithms, a policy of cyclical learning rates, meaning it periodically increases for a step size and then decreases the learning rates, may converge faster and better [25,26]. In addition, the fastai version of the One-cycle Policy comprises three complementary techniques that balance the trade-off between fast convergence and overshooting. The Slanted Triangular Learning Rates (STLR) [10] and the

[1] https://fastai1.fast.ai/text.transform.html#SpacyTokenizer

[2] https://ohmeow.github.io/blurr/modeling-core/#hf_splitter

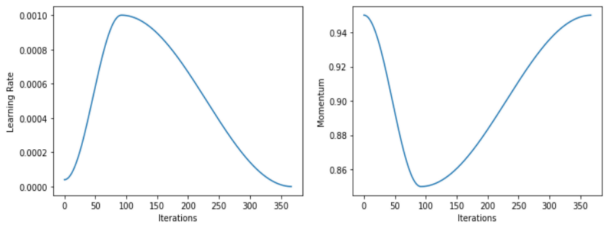

Figure 2: STLR and Cyclical Momentum. Image Credit: [7]

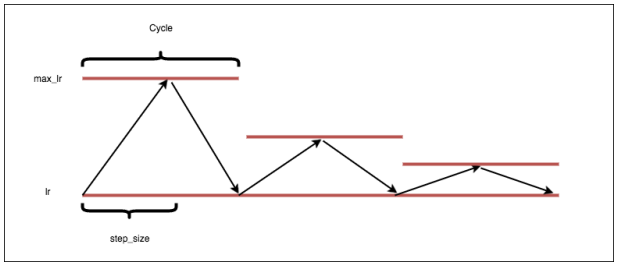

Figure 1: One-cycle Policy with a Max-learning-rate Decay. Image credit: https://github.com/bckenstler/CLR

Cyclical Momentum [25,26] allow us to micro-manage iterations/updates within a cycle, whereas changing maximum learning rate (`max_lr`) per cycle let us control the quality of each. Empirically, STLR and cyclical momentum together work best when they simultaneously change in a reversed direction. As 1 shows, it uses a warm-up and annealing for the learning rate while doing the opposite with the momentum. 2, on the other hand, indicates that we apply a simply decay on `max_lr` per cycle.

2.4.3 Other Optimization Schemes. We test several optimizers and find none of them improve the convergence stability significantly than Adam [13]. For the choice of loss function, we realize that there's no need to use the label smoothing function [22] since the FinNum-2 task is in a tyical binary classification setting.

3 Experiments

In this section we report our setting for the official runs and discuss the results.

3.1 Hyperparameters

For CYUT-1, we run 10 epochs using a batch-size of 32, with the learning rate being 1e-7. Most of Adam optimizer related hyperparameters remain default. For CYUT-2, however, we also apply Mixed Precision to the optimizer, and assign a class weight ratio of 4.28:1 to the loss function. The ratio is simply the inverse of the class distributions. As for the One-cycle scheme specific to CYUT-2, every cycle runs one epoch in a batch-size of 8. All cycles share the same range of Cyclical Momentum, which uses the default of fastai. For the step size of STLR, we also simply let fastai decide it. In the following, we list the most important configurations of discriminative fine-tuning, i.e., the ranges of the learning rates and their decays among cycles:

1. 3 cycles: 5e-4 – 5e-7
2. 1 cycle: 5e-5 – 5e-8
3. 1 cycle: 1e-8 – 1e-5

One may notice that the learning rate range is always a factor of 1000, such that the four layer-groups may roughly have the rates distributed evenly. However, it comes to our attention that, after the timing of the official runs, the version 3.3.0 and above of HuggingFace's Transformers has removed the pooling layer, because in theory they are unrelated to classification. Should any reader want to reproduce the outcome, please be advised that it will definitely vary if using a more recent version of HuggingFace's Transformers.

3.2 Official Run and Additional Run Results

Table 1 shows the official run results in formal run. Since we find that we made a mistake in our first run, we made additional runs after we fixed our bug. The additional run results are shown in Table 2. The discussions of our first run will be based on the additional runs.

3.3 Run 1 Settings and Discussions

The performance in the CYUT-1 w/o preprocessing in Table 2 can be seen as the performance of BERT in this problem. The F-1 scores are about 49%. The same model can be improved greatly by our preprocessing strategy. The F-1 score is 86.6% for the development set and 62.7% for the test set. The performance of formal test drops greatly compares to the development test. We believe that this is caused by the different distribution of the data sets, since the average performance drop 24% as shown in the last row in Table 1.

3.4 Run 2 Settings and Discussions

The performance in the CYUT-2 is shown in Table 1. The F-1 score is 95.99% for the development set and 71.90% for the test set. The

Table 1: Official Run Experiment Results (macro F-1 in %) [5]

	Development	Test
Majority	44.88	44.93
CYUT-1	48.64	48.02
CYUT-2	95.99	71.90
Average of 17 runs	88.18	64.11

Table 2: Additional Run Experiment Results (macro F-1 in %)

	Development	Test
CYUT-1 +preprocessing	86.6	62.7
CYUT-1	49.9	49.2

performance of formal test also drops greatly. According to the overview report, the test result ranks second best [5]. The performance of formal test also drops greatly compares to the development test.

For the performance gap, we notice that the false negatives are somewhat more frequent than the false positives. We wonder whether it means the model of CYUT-2 overfits due to the class weights, or the model simply don't find any clue for the positives. We skim read some false negatives and find an intriguing yet probably representative case of a numeral "2C." In the test set, a tweet uses to refer the link between global warming and the stock price of Tesla. In the training and the development sets, however, all of the "2C" and "2c" stand for "to see." This case indicates that, both informal usages of tweet and the domain knowledge of stocks can use some more efforts.

4. Related Works

As aforementioned in the sub-section of fine-tuning techniques, we have conducted preliminary tests of optimization algorithms and loss functions. Due to the long training time a deep neural network needs, Layer-wise Adaptive Rate Scaling (LARS) [34] aims to implicitly adapt various learning rates for different layers of convolutional networks, and then evolves to amend its shortcoming on BERT with a new layerwise adaptive large batch optimization technique called LAMB [35]. As the name suggests, however, they are designed for relatively big size of batches for the efficiency of pretraining, we fail to find significant improvements using them for fine-tuning. The fact that we're already using discriminative fine-tuning may further complicate the behavior of convergence.

Another perspective on taming the behavior of convergence is about stabilizing gradient updates. Lookahead [36], Rectified Adam [18], and Gradient Centralization [33] fall into this category. Ranger[3] further combines them together as one optimizer. Again, based on our trials for the FinNum-2 task, they are neither more effective nor more efficient.

Last but not least, if we see the tokenization tricks as feature engineering for deep neural networks, while being seldom used for text classification and fine-tuning, it is actually a common approach for text generation and pretraining. CTRL [12] and GPT-3 [2] have many designated "prompts" that enable conditioned generations. Feature Engineering done in such a preprocessing manner may be easier for adapting different tasks or pretrained models than specialized embeddings.

5. Conclusion and Future Works

In this paper, we report our approaches to the FinNum-2 shared task in NTCIR-15, we built our systems based on BERT model and XLM-RoBERTa model. In our CYUT-2 run, the F-1 score is 95.99% for the development set and 71.90% for the test set. The test result ranks second best among all participants. It is notable that in both of our fixed runs and official runs, the performance of formal test drops greatly compares to the development test. We believe that this is caused by the different distribution of the data sets. In the future, we shall find some ways to fixed this gape. For example, the difference might be caused by the new terms, then the contemporary data might be necessary for training. On the other hand, the language usage might be changed, then more user data might be added into the training set.

ACKNOWLEDGMENTS

This study was supported by the Ministry of Science and Technology under the grant number MOST 109-2221-E-324-024.

REFERENCES

[1] Abderrahim Ait Azzi and Houda Bouamor. 2019. Fortia1@ the NTCIR-14 FinNum task: enriched sequence labeling for numeral classification. (2019).

[2] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and others. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165* (2020).

[3] Chung-Chi Chen, Hen-Hsen Huang, Hiroya Takamura, and Hsin-Hsi Chen. 2019. Overview of the NTCIR-14 FinNum Task: Fine-grained Numeral Understanding in Financial Social Media Data. In *Proceedings of the 14th NTCIR Conference on Evaluation of Information Access Technologies*, 19–27.

[4] Chung-Chi Chen, Hen-Hsen Huang, Hiroya Takamura, and Hsin-Hsi Chen. 2019. Final Report of the NTCIR-14 FinNum Task: Challenges and Current Status of Fine-Grained Numeral Understanding in Financial Social Media Data. In *NII Conference on Testbeds and Community for Information Access Research*, Springer, 183–192.

[5] Chung-Chi Chen, Hen-Hsen Huang, Hiroya Takamura, and Hsin-Hsi Chen. 2020. Overview of the NTCIR-15 FinNum-2 Task: Numeral Attachment in Financial Tweets. In *Proceedings of the 15th NTCIR Conference on Evaluation of Information Access Technologies*, 0–0.

[6] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised Cross-lingual Representation Learning at Scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, Online, 8440–8451. DOI:https://doi.org/10.18653/v1/2020.acl-main.747

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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)*, Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186. DOI:https://doi.org/10.18653/v1/N19-1423

[8] Jeremy Howard and Sylvain Gugger. 2020. Fastai: A Layered API for Deep Learning. *Information* 11, 2 (February 2020), 108. DOI:https://doi.org/10.3390/info11020108

[9] Jeremy Howard, Sylvain Gugger, Soumith Chintala, and an O'Reilly Media Company Safari. 2020. *Deep learning for coders with fastai and PyTorch: AI applications without a PhD*.

[10] Jeremy Howard and Sebastian Ruder. 2018. Universal Language Model Fine-tuning for Text Classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Association for Computational Linguistics, Melbourne, Australia, 328–339. DOI:https://doi.org/10.18653/v1/P18-1031

[11] Hairong Huo and Mizuho Iwaihara. 2020. Utilizing BERT Pretrained Models with Various Fine-Tune Methods for Subjectivity Detection. In *Web and Big Data*, Springer International Publishing, Cham, 270–284.

[12] Nitish Shirish Keskar, Bryan McCann, Lav Varshney, Caiming Xiong, and Richard Socher. 2019. CTRL - A Conditional Transformer Language Model for Controllable Generation. *arXiv preprint arXiv:1909.05858* (2019).

[13] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Retrieved from http://arxiv.org/abs/1412.6980

[14] Taku Kudo. 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*

[3] https://github.com/lessw2020/Ranger-Deep-Learning-Optimizer

1: Long Papers), Association for Computational Linguistics, 66–75. Retrieved from http://aclweb.org/anthology/P18-1007

[15] Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 66–71.

[16] Guillaume Lample and Alexis Conneau. 2019. Cross-lingual Language Model Pretraining. *Advances in Neural Information Processing Systems (NeurIPS)* (2019).

[17] Chao-Chun Liang and Keh-Yih Su. 2019. ASNLU at the NTCIR-14 FinNum task: incorporating knowledge into DNN for financial numeral classification. In *Proceedings of the 14th NTCIR Conference on Evaluation of Information Access Technologies*.

[18] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. 2020. On the Variance of the Adaptive Learning Rate and Beyond. In *International Conference on Learning Representations*. Retrieved from https://openreview.net/forum?id=rkgz2aEKDr

[19] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *CoRR* abs/1907.11692, (2019). Retrieved from http://arxiv.org/abs/1907.11692

[20] Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018. Regularizing and Optimizing LSTM Language Models. In *International Conference on Learning Representations*. Retrieved from https://openreview.net/forum?id=SyyGPP0TZ

[21] Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018. An Analysis of Neural Language Modeling at Multiple Scales. *CoRR* abs/1803.08240, (2018). Retrieved from http://arxiv.org/abs/1803.08240

[22] Rafael Müller, Simon Kornblith, and Geoffrey E Hinton. 2019. When does label smoothing help? In *Advances in Neural Information Processing Systems*, 4694–4703.

[23] Matthew E. Peters, Sebastian Ruder, and Noah A. Smith. 2019. To Tune or Not to Tune? Adapting Pretrained Representations to Diverse Tasks. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, Association for Computational Linguistics, Florence, Italy, 7–14. DOI:https://doi.org/10.18653/v1/W19-4302

[24] Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Association for Computational Linguistics, Berlin, Germany, 1715–1725. DOI:https://doi.org/10.18653/v1/P16-1162

[25] Leslie N. Smith. 2017. Cyclical Learning Rates for Training Neural Networks. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 464–472. DOI:https://doi.org/10.1109/WACV.2017.58

[26] Leslie N Smith. 2018. A disciplined approach to neural network hyper-parameters: Part 1 – learning rate, batch size, momentum, and weight decay. *arXiv.org* (2018). Retrieved from http://arxiv.org/abs/1803.09820

[27] Alan Spark. BRNIR at the NTCIR-14 finnum task: Scalable feature extraction technique for number classification⋆.

[28] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, \Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All You Need. In *Advances in neural information processing systems*, 5998–6008.

[29] Wei Wang, Maofu Liu, Yukun Zhang, Junyi Xiang, and Ruibin Mao. 2019. Financial Numeral Classification Model Based on BERT. In *NII Testbeds and Community for Information Access Research - 14th International Conference, NTCIR 2019, Tokyo, Japan, June 10-13, 2019, Revised Selected Papers* (Lecture Notes in Computer Science), Springer, 193–204. DOI:https://doi.org/10.1007/978-3-030-36805-0_15

[30] Wei Wang, Maofu Liu, and Zhenlian Zhang. 2019. WUST at the NTCIR-14 FinNum Task. In *Proceedings of the 14th NTCIR Conference on Evaluation of Information Access Technologies*.

[31] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. HuggingFace's Transformers: State-of-the-art Natural Language Processing. *CoRR* abs/1910.03771, (2019). Retrieved from http://arxiv.org/abs/1910.03771

[32] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *CoRR* abs/1609.08144, (2016). Retrieved from http://arxiv.org/abs/1609.08144

[33] Hongwei Yong, Jianqiang Huang, Xiansheng Hua, and Lei Zhang. 2020. *Gradient Centralization: A New Optimization Technique for Deep Neural Networks*.

[34] Yang You, Igor Gitman, and Boris Ginsburg. 2017. Large batch training of convolutional networks. *arXiv preprint arXiv:1708.03888* (2017).

[35] Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. 2020. Large Batch Optimization for Deep Learning: Training BERT in 76 minutes. In *International Conference on Learning Representations*. Retrieved from https://openreview.net/forum?id=Syx4wnEtvH

[36] Michael Zhang, James Lucas, Jimmy Ba, and Geoffrey E Hinton. 2019. Lookahead optimizer: k steps forward, 1 step back. In *Advances in Neural Information Processing Systems*, 9597–9608.