Training Experimental Language Models with Low Resources, for the Hungarian Language

Zijian Győző Yang[1,2] **and Tamás Váradi**[1]

[1]Hungarian Research Centre for Linguistics,
Benczúr u. 33, H-1068 Budapest, Hungary
yang.zijian.gyozo@nytud.hu; varadi.tamas@nytud.hu

[2]MTA-PPKE Hungarian Language Technology Research Group
Práter u. 50/A, H-1083 Budapest, Hungary
yang.zijian.gyozo@itk.ppke.hu

Abstract: In recent years, natural language processing tasks, like sentiment analysis, can be solved with high performance techniques, if a pre-trained language model is fine-tuned. However, in most cases, the pre-training of language models require huge computational resources and training corpora. Our paper addresses the issue of developing deep neural network language models for low resourced languages, such as Hungarian. Pre-training language models like BERT, requires a prohibitive amount of computational power and huge amount of training data. Unfortunately, neither of these prerequisites are commonly available for low resource languages. The question is how well the system can perform with limited resources (both in data and hardware). We focus our research on five transformer models: ELECTRA, ELECTRIC, RoBERTa, BART and GPT-2. To evaluate our models, we fine-tuned the models in six different natural language processing tasks: sentence-level sentiment analysis, named entity recognition, noun phrase chunking, extractive summarization and abstractive summarization. Our results suggest that while our experimental models obviously cannot surpass the performance of the state-of-the-art Hungarian BERT model, they require a smaller carbon footprint, may bring neural network technology to mobile applications and, finally, they may lower the threshold to engaging with neural network technology in low resourced languages, which has been an obstacle so far, in the synergistic co-development of cognitive info-communication systems and its related disciplines.

Keywords: ELECTRA; ELECTRIC; RoBERTa; BART; GPT-2; sentiment analysis; named entity recognition; noun phrase chunking; text summarization

1 Introduction

Natural languages provide an important platform for thought and communication, which are considered as pivotal human cognitive characteristics. Therefore, natural

language processing can provide valuable insights into human cognitive processes [1]. Communication between a human and an artificial cognitive system is called inter-cognitive communication (information transfer occurs between two cognitive beings with different cognitive capabilities) [2] [3]. Machine Learning methods are vital elements of modern cognitive info-communication systems because they can be used in various ways such as behavior modeling or sentiment analysis [4].

Modern cognitive science aims to develop a general theoretical framework that encompasses the entire set of human mental capabilities including language. Several attempts have been made to implement human cognitive processes as computer algorithms, for example by using artificial intelligence to explain various aspects of language acquisition and processing [5]. Thus, we can refer to these artificial intelligence-based solutions not merely as language models, but as cognitive models as well.

Natural language processing (NLP) has seen spectacular progress with the application of neural network technology, in particular, the Transformer model [6]. The first breakthrough was the word embedding method [7], which represents words as multi-dimensional continuous vectors. The vector representation of the words creates a multidimensional semantic space in which words with similar meaning are located close to one another. Alongside their semantic content, the word vectors can be trained to incorporate syntactic features, as well. A major drawback of this method was that the vectors were computed on word forms and the system was not able to handle wordforms which had different or even unrelated meanings. In order to solve this issue, alternative models were created, which are based on contextual embedding, for instance ELMo [8], BERT [9], or derivatives of BERT (for example RoBERTa [10]), in which the word vectors reflect context-dependent variation in meaning. As noted above, one major limitation of building neural network-based language models is that it requires extreme amountnumber of resources in terms of data and computational power. In order to overcome these previously described limitations Google Inc. developed the ELECTRA model [11], which is able to achieve similar results to the traditional BERT models by using less resources (1 GPU) and under less time. Furthermore, the size of the trained models is considerably smaller as well, which is important in our current ‘mobile’ world. But ELECTRA cannot solve every kind of natural language processing task, for instance text generation tasks. Thus, we also performed experiments with other types of language models in this research. We also conducted experiments with RoBERTa, BART and GPT model architectures as well. In recent years, a lot of interest has been developed towards the scientific investigation of human-computer interaction. The extremely fast pace of infocommunication device development and the rapidly growing number of device users, hence the increasing occurrence of human-machine interactions renders it especially important to systematically address the challenges that might arise during such encounters. Cognitive Info-communication can greatly benefit from the advancements and innovative solutions developed by cognitive linguistics. Furthermore, several important new models

supported by artificial intelligence could further elevate the successful outcome of these synergistic approaches [2]. Therefore, we aim to contribute to these ground-breaking initiatives by presenting our low-resource applications.

BERT (abbreviation of Bidirectional Encoder Representations from Transformer) is defined as a multi-level, bidirectional Transformer encoder [6]. The BERT model is pre-trained on two language modeling tasks: word masking and next sentence prediction. In the course of the masking procedure 15% of the words in the training corpus is randomly masked and the system has to guess the masked words. In the next sentence prediction tasktask, the system is to guess whether two sentences are consecutive in relation to one another in the text or just randomly chosen. In order to drastically reduce the size of the dictionary, words are broken into pieces on statistical grounds (what is called 'wordpieces') using a tokenizer algorithm [12]. Following the initial training of the BERT, the pre-trained model is fine-tuned in order to be optimal for a certain task.

The first native BERT model in Hungarian was published by Dávid Márk Nemeskey [13], named huBERT. Three different huBERT models were created:

- huBERT: BERT base trained on Hungarian Web Corpus 2.0 [14]
- huBERT Wikipedia cased: BERT base trained on Hungarian Wikipedia without uppercase to lowercase transformation
- huBERT Wikipedia lowercased: BERT base trained on Hungarian Wikipedia with uppercase to lowercase transformation

The huBERT base pre-trained on Hungarian Web Corpus reached remarkable results in such tasks as name entity recognition and noun phrase identification [15]. Feldmann et al. have built the first BERT large model for Hungarian [16].

The models and scripts are can be found in our Github sites [17] [18].

2 Experimental Models

2.1 ELECTRA

The ELECTRA [11] is based on the GAN (Generative Adversial Network) [19] method. The basis of this methodology is that two networks are trained: one is the generator and the other is the discriminator. In the course of pre-training the generator randomly creates vector representations, from which its output is derived. Next, a real output is presented to the generator, and based on this it optimizes the random vector generation process. Consequently, the generator becomes 'smarter' by the end of the training it will be able to generate an output that is similar to the

real one. Meanwhile, the discriminator is trained to decide whether a given dataset is true or false. In order to achieve this, real datasets from the training corpus and generator-derived datasets are both presented to the discriminator. As opposed to the word-masking approach of the BERT, here the task of the system is not to decide what the masked word was, but rather whether the word in question is the original one or exchanged. The two networks are interconnected and their function is facilitated by one another during training. The ELECTRA employs the GAN method in order to train language models. It is differentiated from the BERT model in that the network does not predict the masked words, rather the generator creates words corresponding to masked words, then the discriminator is trained to decide whether the word derived from the generator is original or randomly generated. Therefore, the generator is trained which words are the best-fitting in relation to the masked words, while the discriminator is trained to decide whether the words from a given text input are actually real or not. After the training the generator is discarded and only the discriminator is used further for the fine-tuning.

Google Inc. developed three different ELECTRA models:

ELECTRA small: **12 layers; hidden layer size: 256; 14M parameters**

ELECTRA base: **12 layers; hidden layer size: 768; 110M parameters**

ELECTRA large: **24 layers; hidden layer size:1024; 335M parameters**

The ELECTRA small model requires the least amountnumber of resources, therefore in our research we only did experiments with ELECTRA small models.

2.2 ELECTRIC

Electric was developed as an implementation of the cloze task using an energy-based model [20]. As its name suggests, it features high similarity to ELECTRA models. Contrary to BERT models, Electric does not use masking or a softmax-based normalization. Instead, Electric assigns an energy score to each token position and calculates the distribution of possible tokens. During the training of Electric Noise-Contrastive Estimation is applied. Two key differences can be pointed out between ELECTRA and Electric models. The first major distinction is how the models approach noise distribution: ELECTRA uses a masking algorithm, while Electric applies a two-tower cloze model [21], which uses the context to both sides of the tokens. This involves the application of two transformers, one operates left-to-right and the other processes the sequence right-to-left. Another important difference between ELECTRA and Electric models is that ELECTRA calculates likelihood scores only for the masked tokens, while in the case of Electric these scores can be computed simultaneously for all input tokens. However, a disadvantage of Electric over ELECTRA is the apparent inflexibility of Electric in the choice of noise distribution.

2.3 RoBERTa

The RoBERTa [10] follows the pre-training procedure of BERT, but the following modifications are introduced to enhance the performance of the model:

- Bigger size of the pre-training corpus: In order to train RoBERTa, the size of the corpus is multiplied 10 times, e.g., 160 GB data which comes from 5 different corpora and consists of 30 billion words.
- Longer model training step: Experimentation was done with 100 thousand, 300 thousand and 500 thousand steps. The results indicate that including more steps increases the system performance.
- Bigger batch size: Experiments have been done applying a batch size of 256, 2000 and 8000. It was shown that higher batch size leads to better results. The best performance is achieved using a batch size of 8000. The number of training steps and the batch size are interdependent, a higher batch size requires less training steps in order to achieve the same outcome.
- Exclusion of the Next Sentence Prediction (NSP) task from the pipeline: According to experimentally-concluded claims of the authors, the NSP task does not contribute significantly to the training of the system, therefore they eliminated this particular task.
- Longer input texts: The RoBERTa takes maximal advantage of the sequence length of 512. The sequences do not contain a single sentence only with the rest of the sequence padded by the empty character <pad>, but, rather, they are filled by multiple sentences until the complete 512 character-long sequence is reached. Even the end of document would not mean new sequence opening, in that case a document separator is inserted followed by the input of the rest of the text.
- Dynamic masking: Static masking is applied in BERT, in which the pre-processing of the text involves masking 15% of the text, and this masked 15% stays identical during the pre-training process. As against this, RoBERTa applies dynamic word masking, which means that the word masking pattern is always reestablished before the sequence is presented to the system.
- BPE coding: The RoBERTa encodes the internal representation of the words using the Byte Per Encoding (BPE) method [22], which is a hybrid of word and character representation. Word entities result from the iterative unification of character n-grams, and these are based not on UNICODE characters, but bytes instead.

Three different RoBERTa models can be trained:

RoBERTa small: 6 layers; hidden layer size: 768; 65M parameters

RoBERTa base: 12 layers; hidden layer size: 768; 125M parameters

RoBERTa large: 24 layers; hidden layer size: 1024; 355M parameters

In our research we did experiments with a RoBERTa small model.

2.4 BART

BART [23] is a Transformer-based denoising autoencoder that can be used for the pretraining of sequence-to-sequence models. It has an encoder-decoder architecture. It uses a 'noised' source text as input, then it reconstructs the original text by predicting the corrupted parts. BART has a similar setup to BERT [9], however, with characteristic differences in its architecture. Notably, one such difference is the additional cross-attention of the layers over the final hidden layer, which is present in BART, but not in BERT. The application of BART offers a high degree of flexibility regarding the usage of noising schemes, which is illustrated by the fact that any type of document corruption is compatible with the system as opposed to other denoising autoencoders. BART practically combines a BERT type model with a GPT type model. The difference from BERT model is that the denoising tasks are different:

- Token Masking: random tokens are sampled and replaced with <mask> elements.
- Deletion: random tokens are deleted from the input.
- Text Infilling: number of text spans [24] are sampled, with span lengths drawn from a Poisson distribution. Each span is replaced with a single <mask> token.
- Sentence Permutation: a document is divided into sentences based on full stops, and these sentences are shuffled in a random order.
- Document Rotation: a token is chosen uniformly at random, and the document is rotated so that it begins with that token.

BART outperforms all previously established models in summarization tasks. Two different BART models can be trained:

BART base: 12 layers; hidden layer size: 768; 139M parameters

BART large: 24 layers; hidden layer size: 1024; 406M parameters

We conducted experiments with a BART base model.

2.5 GPT Models

Natural Language Processing can greatly profit from the advancement in the area of generative model pre-training (abbreviated as GPT), which has a decoder-only architecture. The OpenAI research group put the emphasis on a level of semantic investigation that is higher than at the level of words, which facilitates the vector representation of higher-level text units. Furthermore, the application of unsupervised pre-training can help to capture linguistic information which can be

extended to long-term extraction of information by choosing the right transformer net. Additionally, using auxiliary training objectives can increase performance as it is showcased in the work by Rei et al. [25]. Language models can be trained to solve language processing tasks without supervision if the training is performed on large datasets, for example WebText that includes the content from millions of websites. GPT-2 [22], which is a transformer model comprising 1.5 billion parameters reached best-in-class performance in 7 out of 8 tasks in a zero-shot setting. The GPT-3 model [26] improves model performance by the simultaneous increase of parameters and computational capacity. The model has 175 billion parameters and it is capable of achieving state-of-the-art results in several tasks without any fine-tuning. For the efficient supervised training of language models, it is crucial to have large datasets that are not annotated. In previous publications several corpora were used to train language models, for example Wikipedia, Gigaword [27], or the non-public Google News corpus, the RealNews database [28]. GPT models use BPE coding for their dictionary. The current 3+1 GPT models in comparison:

- **GPT:** 12 layers, 12 attention heads; 768 word embedding size; 512 text length; 117 million parameters
- **GPT-2:** 48 layers, 12 attention heads; 1600 word embedding size; 1024 text length; 1.5 billion parameters
- **GPT-3:** 96 layers, 96 attention heads; 12888 word embedding size; 2048 text length; 175 billion parameters
- **GPT Neo (Brown et al., 2021):** mesh-tensorflow library implementation in order to train GPT-3 type models

In our research we performed experiments with the GPT-2 model.

3 Corpora

3.1 Pre-Training Corpora

The GPT-2 model was trained with the Hungarian Wikipedia, which is sequenced to paragraphs [14]. In order to train the ELECTRA, ELECTRIC, RoBERTa and BART models, three different corpora were used:

- **Hungarian Wikipedia (wiki) [14]:** 13,098,808 segments (sentences); 163,772,783 tokens. Used for training ELECTRA, ELECTRIC, RoBERTa, BART and GPT-2
- **NYTK corpus v1 (nytk) [16]:** 283,099,534 segments (sentences); 3,993,873,992 tokens. Used for training ELECTRA and ELECTRIC

- **Webcorpus 2.0 (web) [14]:** 100,255,504 segments (paragraphs); 9,095,424,717 tokens. Used for training BART

Vocabulary sizes:

- The size of the vocabulary that was used to train the ELECTRA 31 and the ELECTRIC models: 31.101
- The size of the vocabulary that was used to train the ELECTRA 64 models: 64.000
- The size of the vocabulary that was used to train the RoBERTa and the BART models: 30,000
- The size of the vocabulary that was used to train the GPT-2 model: 33,000

In the case of ELECTRA and ELECTRIC projects, since they are BERT-based models, we used the two vocabulary files that were created for the HILBERT model. RoBERTa and BART use the same vocabulary, which was trained on the Hungarian Wikipedia. In the case of GPT-2, we conducted two different experiments. In the first experiment, we used the same vocabulary with as RoBERTa and BART, while in the second experiment, we used a new vocabulary with a size of 33,00 that was trained on Hungarian Wikipedia and Webcorpus 2.0. As the second dictionary was trained from a larger corpus, we decided to utilize this dictionary in our current research.

3.2 Fine-Tuning Corpora

The following corpora were used to train the models (more information on the size of the corpora can be found in Table 1):

- **SENT**: the Hungarian Twitter Sentiment Corpus [29] was used for sentence-level sentiment analysis purposes that is created by Precognox [30]. According to the international benchmarks [31] we created three subcorpora from this corpus:
  - **2-class**: binary classification subcorpus. We converted the scores 1 and 2 to 0 as negative, scores 4 and 5 to 1 as positive. Score 3 was ignored to avoid the ambiguities. Training corpus: 2,468 segments. Test corpus: 269 segments.
  - **3-class**: We converted the score 1 and 2 to negative, score 3 to neutral and scores 4 and 5 to positive. 3,600 segments. Test corpus: 400 segments.
  - **5-class**: original five-point likert scaled corpus. 1: very negative, 2: negative, 3: neutral, 4: positive, 5: very positive. Training corpus: 3,600 segments. Test corpus: 400 segments.

- **NER**: As for the fine-tuning of the name recognition (NER) the Szeged NER (SzNer) corpus [32] and NYTK-NerKor (NerKor) corpus [33] were used.
- **NP**: for the noun group recognition (NP) the Szeged Treebank [34] was used. In order to ensure comparability, we used the same corpora for the NER and NP fine-tuning, such as emBERT [15].
- **SUM**: For the summarization task, we used the H+I corpus that Yang et al. used in their research [35]. Training corpus: 559,162 segments, Test corpus: 3,000 segments.

Table 1
Attributes of the fine-tuning corpora

	Segment	**Token #**	**Type #**	**Avg. token #**
H+I	562,162	147,099,485 (art) 16,699,600 (lead)	2,949,173 (art) 749,586 (lead)	263.07 (art) 29.87 (lead)
NerKor	67,524	1,028,114	128,168	15.26
SzNer	9,930	214,096	28,749	22.71
NP	82,097	1,459,227	154,254	17.78
SENT	4,000	59,997	18,423	14.99

4 Experiments

4.1 Pre-Training Experiments

Most of our pre-trained models has a HIL prefix, which is used in reference to the Hungarian Intelligent Language Applications Consortium [18]. One model and some corpora have NYTK prefix, which is used in reference to the Hungarian Research Centre for Linguistics (abbreviation of Nyelvtudományi Kutatóközpont). In this research, we have trained five models with different architectures: ELECTRA, ELECTRIC, RoBERTa, BART and GPT-2.

In order to train the ELECTRA and ELECTRIC models, we used the code implemented by Google Inc. [36] The following two models were trained:

- **HIL-ELECTRA 64 wiki**: ELECTRA small trained on Hungarian Wikipedia, the duration of the training was approximately 5 days. Vocabulary size: 64,000.
- **HIL-ELECTRA 64 nytk**: ELECTRA small trained on nytk corpus, the duration of the training was approximately 7 days. Vocabulary size: 64,000
- **HIL-ELECTRA 31 nytk**: ELECTRA small trained on nytk corpus, the duration of the training was approximately 7 days. Vocabulary size: 31,101.

- **HIL-ELECTRIC nytk**: ELECTRIC small trained on nytk corpus, the duration of the training was approximately 5 days.
- **HIL-ELECTRIC nytk 10%**: ELECTRIC small trained on 10% of nytk corpus, the duration of the training was approximately 0.5 days.

All ELECTRA and ELECTRIC models were trained using one single GeForce RTX 2080 Ti type video card. Training time is affected by the size of the dictionary, the process can be speeded up using a smaller dictionary. We did not modify any parameters during the training process, except for the batch size, which was altered to 80, because higher values were not possible due to size constraints of the CUDA memory.

The training of RoBERTa was performed on a system that contained 4 Nvidia GTX 1080Ti GPU cards, on all 4 GPUs simultaneously. Each GPU had 11 GB memory, thus altogether 44 GB was available for the pre-training. The total duration of the pre-training was 214 hours given the batch size of 8 per card, with a total batch size of 32. The pre-training consisted of 1,250,000 steps, the loss curve decreased from the initial 8.7 value to 2.5, and it got stabilized at around this value. For the pre-training the following hyper-parameters were used: learning rate: 1e-4, epoch: 5, batch size: 8. To train RoBERTa model, we followed the instructions of Huggingface [37]. Only one model was trained:

- **HIL-RoBERTa wiki**: RoBERTa small, trained on Hungarian Wikipedia, the duration of the training was approximately 9 days.

To train the BART model from scratch, we followed the instructions from Huggingface transformers Github [38]. For the pre-training, the following hyper-parameters were used: learning rate: 5e-8, batch size: 8, warmup steps: 10,000. The following two models were trained:

- **HILBART wiki**: BART base trained on Hungarian Wikipedia, the duration of the training was approximately 4 days, epoch: 10.
- **HILBART web**: BART base trained on 10% of Hungarian Webcorpus 2.0, the duration of the training was approximately 4 days.

The loss curve decreased from about the initial 9 value, but never converged. We did experiments with checkpoints where the loss values were between 1 and 2. In the case of BART wiki, we used the checkpoint-250000 (loss: 1.51, eval loss: 1.42). In the case of BART base checkpoint-150000 (loss: 1.14, eval loss: 1.12) were used.

To pretrain the GPT-2 model, we have followed instructions from a TDS article [39]. Only one model was trained:

- **NYTK-GPT-2 wiki**: GPT-2 model trained on Hungarian Wikipedia; the duration of the training was approximately 3 days.

The training of GPT-2 was performed on a system that contained 4 Nvidia GTX 1080Ti GPU cards, with the following hyper-parameters: block size: 100; batch size: 12; buffer size: 1000; learning rate: 3e-5; epoch: 10.

4.2 Fine-Tuning Experiments

In order to test our language models, we performed six different experiments:

1) **Sentence-level sentiment analysis** with 2, 3 and 5 classes. We tested all kinds of our language model on this task.
2) **Name Entity Recognition** (NER). As BART cannot be fine-tuned to token-level classification, only ELECTRA, ELECTRIC, RoBERTa and GPT-2 models were tested on this task.
3) **Maximal Noun Phrase recognition** (NP). As BART cannot be fine-tuned to token-level classification, only ELECTRA, ELECTRIC, RoBERTa and GPT-2 models were tested on this task.
4) **Extractive text summarization**. Since the BertSum tool only compatible with BERT and ELECTRA/ELECTRIC models, only ELECTRA and ELECTRIC models were tested on this task.
5) **Abstractive text summarization**. Since BART and GPT-2 models contain autoregressive decoder, only these models were tested on this task.

For sentence-level sentiment analysis, BART, RoBERTa and GPT-2 were fine-tuned with the code provided by huggingface transformers text classification library [40]. The following modified parameters were used: learning rate = 2e-4, batch size: 32, max sequence length: 128. ELECTRA and ELECTRIC models were fine-tuned with the code provided by Google Inc. [36] The following modified parameters were used: learning rate = 2e-4, batch size: 32, max sequence length: 128. For the best comparison, we have measured the prediction test results on 1-15 epoch numbers. As for the evaluation of sentence-level sentiment analysis models, the *accuracy* method was used.

For NER and NP fine-tuning, ELECTRA and ELECTRIC models were fine-tuned with the code provided by Google Inc. (same as the sentiment analysis). The following modified parameters were used: learning rate = 1e-3; weight decay rate = 0.01, batch size: 4, max sequence length: 128. GPT-2 and RoBERTa models were fine-tuned with the code provided by huggingface transformers token classification library [41]. The following modified parameters were used: learning rate = 1e-4, batch size: 4, max sequence length: 128. For the better comparison with huBERT experiments~\cite{embert}, we used the epoch: 4. As for the evaluation of the NER and NP models, the IOB-based *seqeval* [42] method and *F-measure* were used.

In the case of extractive summarization, the ELECTRA and the ELECTRIC models were fine-tuned with the BertSum tool [43], without any hyper-parameter modification. As for the evaluation of extractive summarization models, the *ROUGE* [44] metric was used.

For abstractive summarization, the BART model was fine-tuned with the code provided by huggingface transformers summarization library [45]. The following modified parameters were used: learning rate: 8e-5, warmup steps: 15,000, fp16, batch size: 8, max source length: 512, max target length: 256, epoch: 20. In the case of GPT-2, in order to fine-tune the models for abstractive summary generation, previously published methodology [22] was applied. The texts from the news articles and the corresponding leads were converted into the following format:

- **1 segment:** [news article text] TL;DR: [lead text]

This corpus was then used to fine-tune the language model facilitated by the pre-trained GPT-2 model. The language model was fine-tuned with the 'language modeling' dictionary [46] provided by Huggingface transformers with the following modified parameters: learning rate: 5e-5: batch size: 4, block size: 512; epoch: 10. As for the evaluation of abstractive summarization models, the *ROUGE* method was used.

5 Results

In the results, for the convenient readability, we excluded the HIL and the NYTK prefixes.

5.1 Text Classification Results

Based on the results presented in Tables 2 and 3, our experimental models could not outperform the state-of-the-art huBERT model. This finding is in line with our expectations, since our models are weaker in some properties (smaller architecture, less training data, smaller batch size, etc.). Despite that we can conclude that our models can achieve considerably high performance. For example, in the sentiment analysis experiments the ELECTRA, ELECTRIC and GPT-2 models are only ~2-7% below the performance of huBERT. It is also important to emphasize that BART and GPT-2 models are well-suited for text generation, as a primary application area.

Table 2
Sentence-level sentiment analysis results

	2-class	**3-class**	**5-class**
huBERT	85.92	72.18	68.50
ELECTRIC nytk 10%	81.85	66.42	63.25
ELECTRIC nytk	82.22	68.92	65.25
ELECTRA 32 nytk	79.55	67.67	60.90
ELECTRA 64 wiki	76.95	62.66	58.40
ELECTRA 64 nytk	77.41	62.91	60.25

RoBERTa	80.00	64.66	61.00
BART wiki	74.07	58.39	54.50
BART web	77.78	60.15	58.25
GPT-2	80.37	63.66	61.00

The Table 3 summarizes the results of NER and NP experiments. In the emBERT [15] experiments, the best results were achieved with 30 epoch, thus in order to assure comparability between different studies, the results of the measurements with 4 epoch were reproduced by us based on the code of the emBERT experiment. In Table 3, as we can see in the NP measurements, the performance of our ELECTRA small models is only 2-3% less compared to that of the Hungarian language BERT (base) model. In spite of the fact that in the majority of the cases the ELECTRA and ELECTRIC small models cannot outperform the BERT models, it is important to emphasize that they can achieve similar performance with fewer parameters, can be trained using 1 GPU and the training lasts under 7 days even with the application of a large dictionary.

Table 3
NER and NP results

	NerKor	**SzNER**	**NP**
huBERT	90.18	97.51	96.97
ELECTRIC nytk 10%	72.84	86.01	90.73
ELECTRIC nytk	78.82	93.63	94.14
ELECTRA 32 nytk	79.34	95.39	94.50
ELECTRA 64 wiki	77.37	94.19	94.14
ELECTRA 64 nytk	77.35	93.59	94.09
RoBERTa	86.04	90.98	94.41
GPT-2	69.43	88.06	85.05

Furthermore, the ELECTRA and ELECTRIC models are customer-friendly in terms of the application of smaller size models, which has proved to be a very important viewpoint. Experiments with ELECTRA showed that decreasing the vocabulary size did not affect the performance of the system, in fact a higher performance was registered. Another noticeable result is that the ELECTRIC 10% used only 10% of corpus and only 0.5 day for training and still achieved comparable performance.

In Figure 1 we can see the evaluation of text classification tasks depending on epoch numbers (we exclude the 3-class and the SzNer experiments). A general finding is that, in the sentiment analysis task, the ELECTRA and ELECTRIC models reach the optimum sooner than the RoBERTa, BART and GPT-2 models. The BART model took the longest to achieve the optimum. It needed more epochs to reach the global optimum. Generally, the BART and GPT-2 models performed less well than other models in most of the tasks. This can be due to the fact that autoregressive models are more suited to text generation tasks.

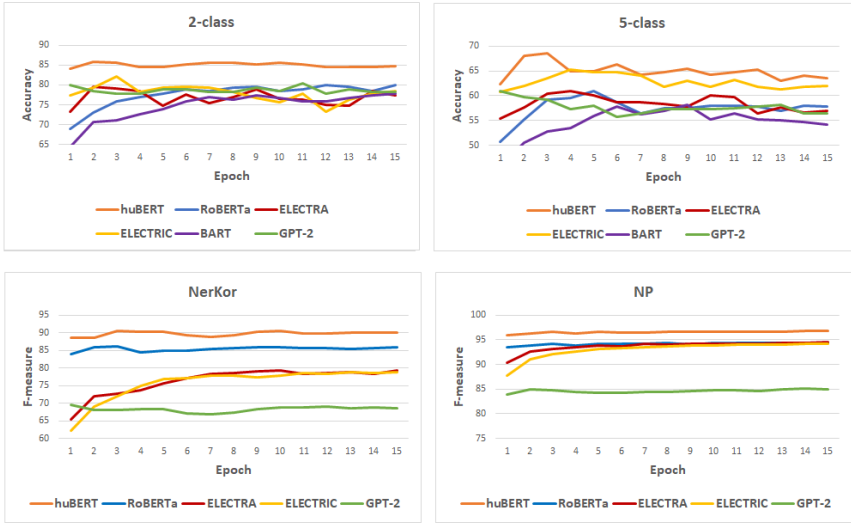

Figure 1
Text classification evaluation

The marginal differences in results highlights the efficiency of transformer models, suggesting that smaller models with less pre-training data can still achieve comparable quality. But in cases where the difference is significant, it indicates that a model pre-trained on a larger resource provides considerable added value.

5.2 Text Summarization Results

To evaluate the summarization tasks, we used the ROUGE-1, ROUGE-2 and ROUGE-L metrics, in the result tables the format is followed:

- ROUGE-1 / ROUGE-2 / ROUGE-L

In Table 4, you can see the results of the extractive summarization task. The huBERT results is from the research of Yang et al. [35].

Table 4
Extractive summarization results

	Recall	**F-measure**
huBERT	49.45/21.07/40.14	27.35/10.78/21.97
ELECTRIC nytk 10%	49.05/20.54/39.77	26.38/10.13/21.16
ELECTRIC nytk	**49.07/20.56/39.79**	**26.40/10.14/21.17**
ELECTRA 31 nytk	49.04/20.53/39.76	26.37/10.12/21.15
ELECTRA 64 wiki	49.02/20.52/39.74	26.36/10.11/21.13
ELECTRA64nytk	48.99/20.51/39.70	26.38/10.13/21.13

For the better comparison, recall and F-measure results are shown in Table 4. As we expected, we could not outperform huBERT, but our models still have comparable results. Our ELECTRIC model is only 0.4% below huBERT. In the competition between ELECTRIC and ELECTRA, the ELECTRIC models won, even the ELECTRIC 10%, which means the architecture of ELECTRIC is more suited to this task than ELECTRA.

In Table 5, you can see the results of the abstractive summarization task. In the research of Yang et al. [35], the PreSumm [47] tool was used. The PreSumm tool support only BERT base models. In their paper, only ROUGE recall results were shown. In the international state-of-the-art research in this field [23], F-measure scores are shown in their results. In Table 5, beside recall, F-measure results are also published. As we can see in Table 5, our BART model could gain significantly higher F-score than the state-of-the-art Hungarian abstractive summarization tool. The recall is lower, but if we analyze the results closer, the summarized text created by tools of Yang et. al. [48] is long. In many cases the summarization is as long as the original article, thus the recall values are so high. In contrast, our BART model also attempts to learn the length of the lead, and it strives to generate shorter summaries.

Table 5
Abstractive summarization results

	Recall	**F-measure**
huBERT	57.07/26.97/48.28	22.42/10.24/18.72
BART	36.49/16.56/27.70	**30.18/13.86/22.92**
GPT-2	40.90/11.89/27.46	23.06/06.56/15.04

In the case of GPT-2, during the test procedure the text to be summarized was input to the system followed by the string "TL;DR:", which served as a trigger that a summary is expected as an output text. The model generated 3 different texts and the first sentence of each generated text was concatenated. These 3 sentences together are considered as the result of the summary generation. This approach is more like PreSumm. Thus, in the case of GPT-2, we can consider the recall results as well. The average length (token number) of the summaries are as follows:

- Original leads: Mean: 26.4, Median: 24
- PreSumm leads: Mean: 104.6, Median: 105
- BART leads: Mean: 28.2, Median: 24
- GPT-2 leads: Mean: 60.1, Median: 58

As we can see from the mean and the median values, the BART model could learn the length of the lead as well. The PreSumm tool generates more than 4 times longer text as lead, thus the recall could be much higher. According to the F-scores, our BART model outperformed the huBERT model in abstractive text generation task.

In the case of GPT-2, our key findings suggest that the summary generation method by GPT-2 can achieve competitive results (using a methodology that is not based on the traditional seq2seq - encoder-decoder type architecture). Interestingly, GPT-2 could even achieve higher ROUGE-1 F-scores than PreSumm, and the recall ROUGE-1 score is higher than the one achieved with BART, but if we analyze the generated outputs, GPT-2 generate much more "hallucination" texts than the other models.

Conclusions

In our research, we built five different kinds of language models for the Hungarian language, with low resources. ELECTRA, ELECTRIC, RoBERTa, BART and GPT-2 models were pre-trained, then to evaluate them, we fine-tuned these models on six different kinds of natural language processing downstream tasks. We tested our models on three kinds of sentence-level sentiment analysis tasks, two name entity recognition tasks, a noun phrase chunking, an extractive summarization task and an abstractive summarization task. In the case of sentiment analysis classification, NER, NP and extractive summarization tasks, as we expected, our models could not outperform the state-of-the-art huBERT models, but they could achieve competitively high results, despite having much fewer parameters and training data. In the case of the abstractive summarization task, our BART model gained significantly ~8% higher ROUGE-1 F-score, over the huBERT-based PreSumm tool.

These results are notable, because we can achieve comparable, competitive or higher results with lower hardware and data resources, which can have two advantages. First, the application of these models results in a smaller carbon footprint by using less computational power, thus, less electricity and second, the smaller models are more user friendly, since they require less space and have a faster loading time. Nevertheless, the reduced complexity of these models can foster enthusiasm towards experimentation with new solutions and this can have a positive effect on accelerating the knowledge transfer between different disciplines, e.g., between computational linguistics and cognitive info-communication.

References

[1] C. Vogel és A. Esposito, „Linguistic and Behaviour Interaction Analysis within Cognitive Infocommunications," in *2019 10th IEEE International Conference on Cognitive Infocommunications (CogInfoCom)*, pp. 47-52, 2019

[2] P. Baranyi és Á. Csapó, „Definition and Synergies of Cognitive Infocommunications," *Acta Polytechnica Hungarica,* Vol. 9, No. 1, pp. 67-83, 2012

[3] P. Baranyi, A. Csapo és G. Sallai, Cognitive Infocommunications (CogInfoCom), Springer International Publishing, 2015

[4] B. Bogdándy, Á. Kovács és Z. Tóth, „Case Study of an On-premise Data Warehouse Configuration," in *2020 11th IEEE International Conference on Cognitive Infocommunications (CogInfoCom)*, pp. 179-184, 2020

[5] C. L. Harris, M. Steyvers és D. C. Atkinsc, „Language and Cognition," 2006

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. u. Kaiser és I. Polosukhin, „Attention is All you Need," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan és R. Garnett, szerk., Curran Associates, Inc., pp. 5998-6008, 2017

[7] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado és J. Dean, „Distributed Representations of Words and Phrases and their Compositionality," in *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pp. 3111-3119, 2013

[8] M. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee és L. Zettlemoyer, „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)*, New Orleans, pp. 2227-2237, 2018

[9] J. Devlin, M.-W. Chang, K. Lee és K. Toutanova, „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)*, Minneapolis, pp. 4171-4186, 2019

[10] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer és V. Stoyanov, „RoBERTa: A Robustly Optimized BERT Pretraining Approach," *CoRR,* 2019

[11] K. Clark, M.-T. Luong, Q. V. Le és C. D. Manning, „ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators," in *ICLR*, 2020

[12] M. Schuster és K. Nakajima, „Japanese and Korean voice search.," in *ICASSP*, pp. 5149-5152, 2012

[13] Dávid Márk Nemeskey, „Introducing huBERT," in *XVII. Conference on Hungarian Computational Linguistics*, Szeged, Hungary, pp. 3-14, 2021

[14] Dávid Márk Nemeskey, „Natural Language Processing Methods for Language Modeling," 2020

[15] Dávid Márk Nemeskey, „Egy emBERT próbáló feladat," (An emBERT-trying Task) in *XVI. Conference on Hungarian Computational Linguistics*, Szeged, pp. 409-418, 2020

[16] Á. Feldmann, R. Hajdu, B. Indig, B. Sass, M. Makrai, I. Mittelholcz, D. Halász, Z. G. Yang és T. Váradi, „HILBERT, magyar nyelvű BERT-large modell tanítása felhő környezetben," (HILBERT, Training Hungarian BERT-large Model in a Cloud Environment) in *XVII. Conference on Hungarian Computational Linguistics*, Szeged, Hungary, pp. 29-36, 2021

[17] Hungarian Research Centre for Linguistics, „Github - Research Group of Language Technology," 2022 [Online] Available: https://github.com/nytud. [Accessed: 09 02 2022]

[18] C. Hungarian Intelligent Language Applications, „HILANCO," 2022. [Online] Available: https://hilanco.github.io [Accessed: 09 02 2022]

[19] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville és Y. Bengio, „Generative Adversarial Nets," in *Advances in Neural Information Processing Systems*, pp. 2672-2680, 2014

[20] K. Clark, M.-T. Luong, Q. V. Le és C. D. Manning, „Pre-Training Transformers as Energy-Based Cloze Models," in *EMNLP*, 2020

[21] A. Baevski, S. Edunov, Y. Liu, L. Zettlemoyer és M. Auli, „Cloze-driven Pretraining of Self-attention Networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Hong Kong, pp. 5360-5369, 2019

[22] A. Radford, J. Wu, R. Child, D. Luan, D. Amodeia és I. Sutskever, „Language models are unsupervised multitask learners," 2019

[23] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov és L. Zettlemoyer, „BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online, pp. 7871-7880, 2020

[24] M. Joshi, D. Chen, Y. Liu, D. S. Weld, L. Zettlemoyer és O. Levy, „SpanBERT: Improving Pre-training by Representing and Predicting Spans," *Transactions of the Association for Computational Linguistics,* Vol. 8, No. 1, pp. 64-77, January 2020

[25] M. Rei, „Semi-supervised Multitask Learning for Sequence Labeling," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vancouver, pp. 2121-2130, 2017

[26] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever és D. Amodei, „Language Models are Few-Shot Learners," in *Advances in Neural Information Processing Systems*, pp. 1877-1901, 2020

[27] D. Graff, J. Kong, K. Chen és K. Maeda, „English gigaword," *Linguistic Data Consortium, Philadelphia,* Vol. 4, No. 1, p. 34, 2003

[28] R. Zellers, A. Holtzman, H. Rashkin, Y. Bisk, A. Farhadi, F. Roesner és Y. Choi, „Defending Against Neural Fake News," in *Advances in Neural Information Processing Systems*, 2019

[29] Precognox, „Hungarian Twitter Sentiment Corpus," 2022 [Online] Available: http://opendata.hu/dataset/hungarian-twitter-sentiment-corpus. [Accessed: 09 02 2022]

[30] Precognox, „Precognox," 2022 [Online] Available: https://www.precognox.hu. [Accessed: 09 02 2022]

[31] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy és S. Bowman, „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*, Brussels, pp. 353-355, 2018

[32] G. Szarvas, R. Farkas és A. Kocsor, „A Multilingual Named Entity Recognition System Using Boosting and C4.5 Decision Tree Learning Algorithms," in *Discovery Science*, Berlin, 2006

[33] E. Simon és N. Vadász, „Introducing NYTK-NerKor, A Gold Standard Hungarian Named Entity Annotated Corpus," in *Text, Speech, and Dialogue - 24th International Conference, TSD 2021, Olomouc, Czech Republic, September 6-9, 2021, Proceedings*, pp. 222-234, 2021

[34] D. Csendes, J. Csirik, T. Gyimóthy és A. Kocsor, „The Szeged Treebank," in *Text, Speech and Dialogue*, Berlin, pp. 123-131, 2005

[35] Z. G. Yang, Á. Agócs, G. Kusper és T. Váradi, „Abstractive text summarization for Hungarian," *Annales Mathematicae et Informaticae,* Vol. 53, No. 1, pp. 299-316, 2021

[36] Google Inc., „Github - google-research/electra," 2021 [Online] Available: https://github.com/google-research/electra. [Accessed: 09 02 2022]

[37] Hugging Face, „Hugging Face," 2020 [Online] Available: https://huggingface.co/blog/how-to-train. [Accessed: 09 02 2022]

[38] Hugging Face, „Github - huggingface/transformers - How to pre-train BART model," 2020 [Online] Available: https://github.com/huggingface/transformers/issues/4151. [Accessed: 09 02 2022]

[39] A. Kayal, „Towards Data Science - Train GPT-2 in your own language," 2020. [Online]. Available: https://towardsdatascience.com/train-gpt-2-in-your-own-language-fc6ad4d60171 [Accessed: 09 02 2022]

[40] Hugging Face, „Github - huggingface/transformers - Text classification examples," 2022 [Online] Available: https://github.com/huggingface/transformers/tree/master/examples/pytorch/text-classification [Accessed: 09 02 2022]

[41] Hugging Face, „Github - huggingface/transformers - Token classification," 2022 [Online] Available: https://github.com/huggingface/transformers/tree/master/examples/pytorch/token-classification [Accessed: 09 02 2022]

[42] H. Nakayama, *seqeval: A Python framework for sequence labeling evaluation,* 2018

[43] Y. Liu, „Github - nlpyang/BertSum," 2019 [Online] Available: https://github.com/nlpyang/BertSum. [Accessed: 09 02 2022]

[44] C.-Y. Lin, „ROUGE: A Package for Automatic Evaluation of Summaries," in *Text Summarization Branches Out*, Barcelona, pp. 74-81, 2004

[45] Hugging Face, „Github - huggingface/transformers - Summarization," 2022 [Online] Available: https://github.com/huggingface/transformers/tree/master/examples/pytorch/summarization. [Accessed: 09 02 2022]

[46] Hugging Face, „Github - huggingface/transformers - Language model training," 2022 [Online] Available: https://github.com/huggingface/transformers/tree/master/examples/pytorch/language-modeling [Accessed: 09 02 2022]

[47] Y. Liu, „Github - nlpyang/PreSumm," 2020 [Online] Available: https://github.com/nlpyang/PreSumm [Accessed: 09 02 2022]

[48] Natural Language Processing Group, „Natural Language Processing Group," 2022 [Online] Available: http://nlpg.itk.ppke.hu/projects/summarize. [Accessed: 09 02 2022]