

Fine-Tuning BERT Models to Extract Named Entities from Archival Finding Aids

Luis Filipe Cunha^{1,*}, José Carlos Ramalho¹

¹*Department of Informatics, University of Minho, Portugal*

Abstract

In recent works, several NER models were developed to extract named entities from Portuguese Archival Finding Aids. In this paper, we are complementing the work already done by presenting a different NER model with a new architecture, Bidirectional Encoding Representation from Transformers (BERT). In order to do so, we used a BERT model that was pre-trained in Portuguese vocabulary and fine-tuned it to our concrete classification problem, NER. In the end, we compared the results obtained with previous architectures. In addition to this model we also developed an annotation tool that uses ML models to speed up the corpora annotation process.

Keywords

Named Entity Recognition, BERT, Web, Corpora Annotation

1. Introduction

In recent works, mechanisms were created to extract and identify named entities in Portuguese archival documents. In [1], several NER models were created using different ML algorithms and architectures in order to study the behavior of these models in the archival domain. Finally, the NER models were made available to the public through a Web platform named NER@DI, presented in [2], which has been updated with new NLP resources and new NER models.

In this paper, we intend to continue the study already done through the research and testing of new ML architectures, namely BERT models that use the transformers architecture. This approach presents a self-learning technique that can efficiently use the GPUs' parallel computational power to pre-train models with hundreds of million parameters, making them have a broader and richer understanding of the language used.

During previous research, we identified another problem in creating Portuguese NER models, i.e., the lack of annotated Portuguese data to train ML models. In fact, generating annotated corpora can become challenging and time-consuming. Thus we created a smart annotator tool that uses our ML models to assist the experimenter in the annotation process.

The intelligent annotator was also made available to the public through NER@DI. In addition, an API was developed and documented, allowing the public to use all the NER models as a service, enabling their integration into other systems.

TPDL2022: 26th International Conference on Theory and Practice of Digital Libraries, 20-23 September 2022, Padua, Italy


*Corresponding author.

✉ filipe-cunha1@hotmail.com (L. F. Cunha); jcr@di.uminho.pt (J. C. Ramalho)

🆔 0000-0003-1365-0080 (L. F. Cunha); 0000-0002-8574-1574 (J. C. Ramalho)



© 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

2. Related Work

In recent years we witnessed an increase in performance of several NLP tasks due to the new research in ML applied to text processing. In [3] Embeddings from Language Models (ELMo) were introduced, presenting a feature-based approach that generates deep contextualized word vectors where the words' representations were functions of all the internal layers of the model. By taking into account not only the word's syntax and semantics but also their context meaning, the same token would have different representations when used in different contexts.

Then, in [4], the authors presented ULMFiT, where they pre-trained a model with a huge dataset containing more than 100 million words [5]. By fine-tuning this pre-trained model, it was reused for different NLP tasks enabling generalization with only 100 labeled examples.

In [6] they created a new architecture, Transformers, with the self-attention mechanism that drastically increased the efficiency of text processing, allowing to create bigger and richer models with higher knowledge of the language vocabulary.

Shortly after that, new models were developed that used the transformers architecture and achieved SOTA results in several NLP tasks, both on sentence and token level, such as BERT [7] and GPT [8].

3. NER@DI

NER@DI is a web platform created to provide various natural language processing tools associated with Named Entity Recognition. This platform was born as a result of a research [1] about the processing of Portuguese archival documents, where several NER models were generated using different ML algorithms and architectures. However, several limitations of this NLP field in Portuguese were identified during this work. So in order to promote research in this area, the NER@DI platform makes the created resources in [1] available to the public, such as Portuguese archival annotated corpora, tools to support the annotation of new corpora, and, of course, the generated ML models.

3.1. Architecture

This platform was created with the intent of being complemented with new features in the future. Thus, a micro-service architecture was used, promoting looser coupling, more flexibility, and portability.

At the moment, it has two containers that correspond to the Data API and the API Gateway.

The API Gateway is implemented with an Nginx web server containing the client application, developed in Vue.js, a framework that uses reactive interfaces. The use of an API Gateway pattern makes the client less coupled to the micro-services, i. e., it does not need to know the internal structure of the server to communicate with the application. The use of this pattern means that there are no direct references between the client and the microservices, so the refactoring and maintenance of these will have a lower impact to the client. On the other hand, if a gateway server was not used, all microservices would be exposed to the public, which could lead to security issues. Then, Vue.js was used to create the client application. It has a small learning curve, so it is fairly approachable, allowing the creation of maintainable interfaces

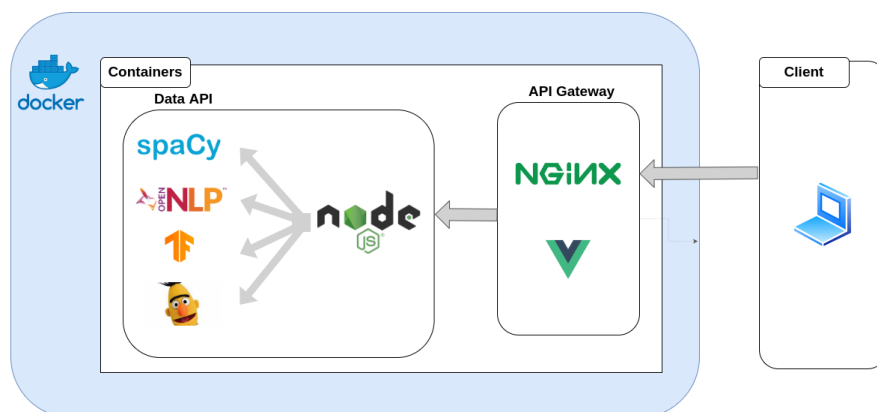


Figure 1: NER@DI architecture.

due to its reusable components mechanism that allows isolating all logic from the views. To complement this framework, Vuetify was used, which consists of a UI library that provides several pre-made reactive components.

The second container is the Data API, which is responsible for receiving, processing and responding to HTTP requests, in this case, associated with the extraction of entities. For this, a Node.js server was used, complemented by [9] library, which works like a broker that is responsible for managing the API routes and delegating the NER processing to the corresponding tools. The Machine Learning models were implemented with OpenNLP, spaCy, Tensorflow and BERT so, to process NER requests, the Node.js server uses the *child_process* [10] library, creating new processes to execute programs in python and java. When the execution of the programs finishes, the created processes return the output to the Node.js server, which is responsible for returning the response to the client in JSON format.

Finally, in order to deploy this platform, each micro-service was wrapped with a docker container. These containers promote the isolation, scalability, agility and portability of each micro-service since it is really easy to install a containerized application in any system that has docker running. At the moment, NER@DI is hosted on the servers of the University of Minho's Informatics Department, at [11].

3.2. Features

During the development of the NER models, several support tools were generated. Some of these tools were selected and implemented in NER@DI in order to make them available to the public. Most of NER@DI features are described in [2], thus, this paper only present the newly added ones:

- Addition of a new model (BERT) trained on Portuguese Archival Finding Aids that allows to perform NER.
- Creation of a documented API that allows NER@DI users to use the NER models as a service in other applications.

- Implementation of a smart annotator that uses ML models to assist the experimenter with the corpora annotating process.

NER@DI can be used by various types of users, for example, historians wishing to extract relevant entities from archival documents or even other developers or researchers with the intent of reusing the annotated datasets in other contexts, or using NER@DI as a service in their own applications.

4. Models

In order to identify and extract entities from natural text, NER@DI uses several ML architectures, such as Maximum Entropy, Convolution Neural Networks (CNN), Bidirectional Long Short-Term Memory with a Conditional Random Field decoder (Bi-LSTM-CRF) and was recently updated with a new model, Bidirectional Encoder Representations from Transformers (BERT).

In NER@DI previous versions, we used NER approaches that consisted of training models from scratch. Now we present a BERT model, which consists of using pre-trained models with hundreds of thousands of parameters and leveraging the knowledge acquired during their pre-train by fine-tuning them on a specific task, in our case, NER. In this paper we will present the BERT model in more detail. The other models were described in [1].

4.1. Bidirectional Encoder Representations from Transformers

By training models from scratch, we randomly initialize the models' weights, which means that the model only learns from the training data. However, what if, instead of creating completely new models, we re-utilize pre-trained models and fine-tune them to our concrete classification task? By doing so we could leverage the knowledge that the model obtained during its pre-train, in a huge amounts of textual data, towards our goal. In practice, we would be using the pre-trained model weights, which would provide the model with a statistical understanding of the language vocabulary, and then fine-tune the model with a task-specific classifier. That's what BERT models do.

Bidirectional Encoder Representations from Transformers (BERT) [7] is a transformer based model focused on the encoder architecture. Since these models are pre-trained with many data, they have large dimensions, reaching billions of parameters, which makes their training process, resource and time expensive. This process requires a large number of GPUs, and can take several days or even weeks consuming high amounts of energy.

In this paper we intended to process Portuguese text so we used BERT models pretrained in Portuguese corpora. First we have BERTimbau [12] with two variants associated with the model size, "bert-base-portuguese-cased" and "bert-large-portuguese-cased" models, with 110M and 330M parameters respectively [12]. These models were trained in a Brazilian Portuguese corpus composed of 2.7 billion [13] tokens. Secondly, there is a multilingual model, "bert-base-multilingual-cased" [14] with 110M parameters, trained on the largest Wikipedia texts, making this model capable of processing texts in 104 different languages, Portuguese included.

Out of curiosity, at the moment, the largest existing model was created by the cooperation between Microsoft and Nvidia, published in October 2021, Megatron-Turing Natural Language

Generation [15], a model with 530 billion parameters trained with 4480 A100 80GB GPUs in a set of 15 datasets consisting of a total of 339 billion tokens.

It is important to note that, by transferring all the pre-trained information to our model, we are increasing its language knowledge, however, all the errors, noise or even bias are also transferred. [16] states that the pre-trained models can be biased to their training data. In this paper, they analysed the GPT-3 model's biases towards gender, race, and religion, however, they state that the model may express other categories of bias. In practice, regarding the religion bias, this behaviour made the model associate words such as "Islam" with "terrorism". As for genre bias, the "female" word was usually associated with tokens such as "naughty" or "beautiful" while the "male" word is associated with the tokens "large", and "lazy".

4.1.1. Sub-word Tokenizer

For an ML statistical model to be able to interpret natural text, it is necessary to transform the text into numerical representations capable of transmitting the meaning of that text to the machine. These representations are the starting point for the ML models, so the higher the amount of information contained in them, the better their interpretation will be, conditioning the performance of the entire ML process. One technique for creating these representations is tokenization.

In NLP, this technique has been studied and widely used in several areas of token processing, such as NER. In this case, word-based tokenizers are usually used, i.e., defining a fixed size N for the vocabulary and then associating an id for the N most frequent words of that vocabulary. This method has shown good results in several contexts, however, it has several limitations. Due to the fact that the number of words is limited, ML models have difficulties dealing with out of vocabulary words or even words that are rarely used. One solution for this problem is to increase the number of vocabulary words (N), however, this would lead to other problems such as making the computational model heavier and increasing the number of rare words. On the other hand, as each distinct word has a different id, similar words have entirely different meanings, which causes information about the words' relationship to be lost during this phase, decreasing the performance of the models.

Thus, in order to solve these limitations and increase the meaning of the numerical representations, models like BERT use a different technique, sub-word tokenization. This method aims to decompose rare words into sub-words, keeping the most frequent words intact. In fact, observing a given vocabulary, words like "tokenization" can be decomposed into two sub-words "token" and "ization". Using a word-based tokenizer, the words "token" and "tokenization" would have representations with entirely different meanings, however, with a sub-word tokenizer, by splitting the word "tokenization" into two sub-words, the model can learn that this word is composed of sub-words that it already knows. In this case, the model associates the word tokenization to the most frequent word "token" thus answering the problem of out-of-vocabulary words and maintaining a vocabulary with a reasonable size [17].

Models that use this method have increased their accuracy, mainly in the classification of unknown words [18].

5. Annotated Data

For the model fine-tuning, we need annotated data so the model learns how to correctly classify the intended named entities. In this work, we used archival finding aids datasets which we had already annotated in [1]. Through various annotation methods, such as the use of regex, ML statistical models, or even manual annotation, nine datasets from three Portuguese archives were annotated, six from the *Arquivo Distrital de Braga*, two from the *Arquivo Regional e Biblioteca Pública da Madeira* and one from the Arquivo Nacional da Torre do Tombo.

In total, the annotated corpora contain 235875 tokens that make up 7397 phrases. As for entities, we annotated more than 36 thousand of entities in total. All the annotated corpora can be downloaded in the NER@DI platform.

6. Data processing

After selecting our models, we started the data loading and transformation process. First, we loaded the training and validation data into memory and then, we started the tokenization process, mapping all the tokens into their corresponding ids in the pre-trained vocabulary. The training samples must comply with the pre-trained vocabulary, having the same format and structure, which means that the tokenizer used for the pre-training must be the same for the fine-tuning. Transformer-based models usually use sub-word tokenizers, meaning that the rarest words of the vocabulary are usually split into sub-words.

Listing 1: Text sequence encoding.

```
# Original
['Pelo', 'qual', 'foram', 'identificados', 'os', 'bachareis', 'que', 'seriam',
 'desembargadores', 'da', 'Relação', 'e', 'Casa', 'do', 'Porto', '.']

# Tokenized
['[CLS]', 'Pelo', 'qual', 'foram', 'identificados', 'os', 'ba', '##char', '##',
 'ei', '##s', 'que', 'seriam', 'desembar', '##gadores', 'da', 'Rela', '##ção',
 'e', 'Casa', 'do', 'Porto', '.', '[SEP]']
```

Even if the training data is already tokenized in IOB format, we have to tokenize it again to create the correct correspondence between the training data and the pre-train vocabulary id. In Listing 1 the tokens "bachareis" and "desembargadores" were divided into smaller sub-tokens. Consequently, since we changed the sequence vectors lengths by splitting tokens into sub-tokens, we also have to remap the labels' vectors to ensure that each token is associated with its correct label. After that, we had to pad and truncate all the sequences according to the pre-trained maximum length parameters.

7. Fine-tuning

With the data processed, we started fine-tuning the model for the archival context. In order to do so, we used the Hugging Face library [19] that allows us to fine-tune several state-of-art NLP models with Keras and Pytorch API.

The fine-tuning process removes the pre-trained model head (last layer) focused on the masked language objective and replaces it with a new randomly initialized head. The idea is to create a new classifier specialized in recognizing entities from archival data, with N outputs, where N is equal to the number of labels.

To fine-tune the model, we only used NVIDIA GEFORCE GTX 1070 Ti GPU. Because of this, we had to adjust the batch size for the bigger models to avoid out-of-memory errors. The fine-tuning of each model lasted from 30 minutes to 2 hours. During this process, we logged the training and evaluation loss in order to generate the model's learning curves.

In Figure 2 we can see the loss values per epoch of the three generated models, bert-large-portuguese, bert-base-portuguese and bert-multilingual. In fact, the models start to overfit between 2 and 4 epochs. In order to retrieve the models with the highest capacity of generalisation, the Huggingface library keeps several model checkpoints during the training phase and selects the best one at the end. Only the best checkpoint will be used for the actual Named Entity Recognition.

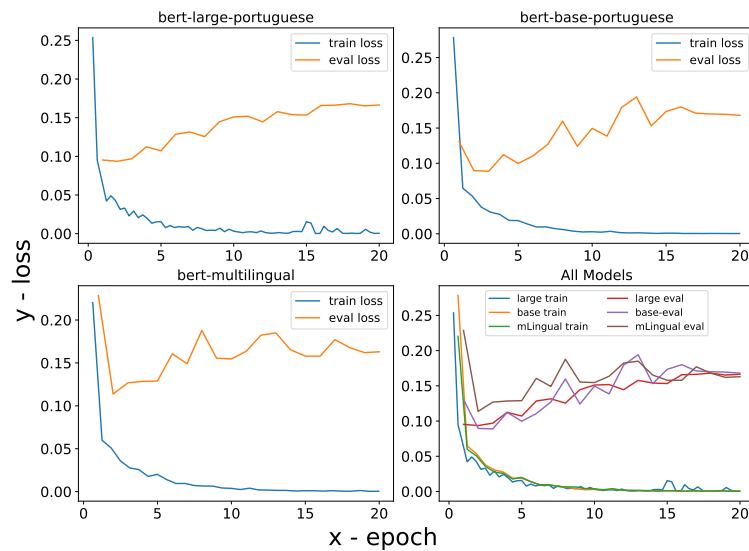


Figure 2: BERT models learning curves.

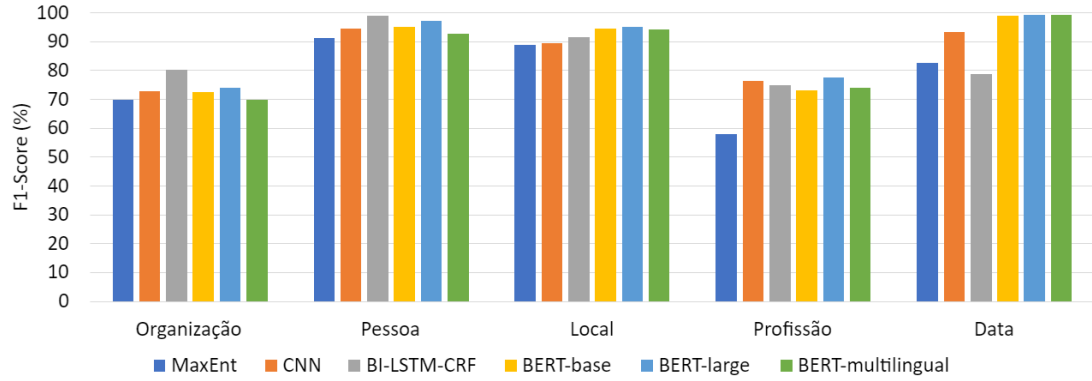
8. Validation Results

In order to fine-tune and validate our NER models we divided the annotated data into two splits, 70% of the data was used for fine-tuning and 30% for validation. The validation results can be seen in the Table 1. Analyzing this Table we can say that the model which obtained the best results was BERT-large achieving an F1-score of 94.53%, followed by BERT-base with 93.19%, and BERT-multilingual with 92.29% F1-score. Despite this, we can say that all the models achieved similar results even though they were pre-trained in different corpora and have different dimensions.

Table 1

Overall models validation results.

Model	Precision(%)	Recall(%)	F1-Score(%)
BERT-base	91.42	95.03	93.19
BERT-large	93.30	95.80	94.53
BERT-multilingual	89.86	94.85	92.29

**Figure 3:** NER results by entity label.

In Figure 3 we have the models' F1-score results per entity type. Here we can analyze and compare the disparity between the entity labels recognition results per model (we also added results from the models trained in [1]). In general, the models demonstrated more difficulty in recognizing Profession and Organization entity types achieving an F1-score between 70% and 80% in those labels. As for the other labels we were able to achieve an F1-score between 90% and 100%.

Then, in Table 2 we validated all the models with a dataset that was not used for training in order to test the models' generalization.

Table 2Models results on *Ruas de Braga* corpus.

Corpus	Model	Precision(%)	Recall(%)	F1-Score(%)
Ruas de Braga	Maxent	73.09	61.09	66.55
	CNN	75.39	62.62	68.42
	BI-LSTM-CRF	50.50	58.80	53.00
	BERT-base	72.37	73.60	72.98
	BERT-large	75.38	74.67	75.03
	BERT-multilingual	72.55	63.70	67.84

The BERT models were able to achieve better results in this test. We can analyze the importance of the model's pre-train. In fact the BERT models were pre-trained with high amount

of Portuguese textual data, giving them the advantage when processing out of vocabulary words and unseen documents.

9. ARCANO - Smart Annotator

The key to obtaining good results in this subfield is the training data quality. The closer the training data is to the data in which this technology is intended to be used, the better the results will be. Good training material is not always available, which creates the need to annotate text. Normally, this activity is performed manually by an experimenter who is knowledgeable about the domain of the documents. However, this task can become time-consuming and tedious despite its low complexity. Thus, the idea of developing an intelligent tool to support text annotation, *ARCANO* [11], was born.

This annotator aims to intelligently assist the entire annotation process using ML models to try to predict named entities of the texts we want to annotate. The idea is to use a generic model to find entities in a small fraction of the target dataset. Then, by correcting the entities found, we teach the model how to classify entities in our specific domain, iteratively.

The sequence diagram in Appendix A illustrates *ARCANO*'s annotation workflow. Initially, the experimenter imports the entire target dataset for it to be annotated. Then, this dataset is divided into N batches that will be used to train the ML model. Firstly, the first batch is sent to the server to identify and classify the entities it can find. Then, the result is sent to the client to be corrected by the experimenter. Now that we have validated training data, we can use it to refine the ML model. Thus, the correctly annotated data is sent to the server to train the model as well as the second batch so that the trained model can extract new entities from it. The more annotated batches, the higher the amount of training data, making the model learn how to classify entities even better. This makes the experimenter annotate fewer and fewer entities manually. The greater the autonomy degree of the model, the lesser the work of the experimenter. Finally, *ARCANO* joins all the previously annotated data into a file, enabling to export the fully annotated dataset.

This tool was used to annotate a corpus from the *Arquivo Nacional da Torre do Tombo*, the *Arquivo de Oliveira Salazar*'s archival finding aids. In total, 71397 tokens were annotated, making more than 7000 different entities. In the end, the annotation process was considerably easier and faster due to the intelligent entity recognition system.

10. Conclusion and Future Work

In previous works we obtained high F1-score results in NER using other ML algorithms, however, the BERT model presented a great advance in the generalization. Using the knowledge obtained during its pre-training, BERT models were able to achieve higher F1-score results in a corpus that was not present in their training data (compared to previous architectures). Furthermore, we also concluded that annotating corpora with the developed annotator was much faster than doing it manually.

For future work, it would be interesting to work on the extracted entities, for example creating disambiguation mechanisms, allowing to relate entities between different documents.

References

- [1] L. F. d. C. Cunha, J. C. Ramalho, Ner in archival finding aids: Extended, Machine Learning and Knowledge Extraction 4 (2022) 42–65. URL: <https://www.mdpi.com/2504-4990/4/1/3>. doi:10.3390/make4010003.
- [2] L. F. Cunha, J. C. Ramalho, Towards entity linking, ner in archival finding aids, in: C. T. Lopes, C. Ribeiro, F. Niccolucci, I. Rodrigues, N. Freire (Eds.), Proceedings of Linked Archives International Workshop 2021 co-located with 25th International Conference on Theory and Practice of Digital Libraries (TPDL 2021), 2021, pp. 22–29. URL: http://ceur-ws.org/Vol-3019/LinkedArchives_2021_paper_12.pdf.
- [3] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, L. Zettlemoyer, Deep contextualized word representations, volume 1, 2018. doi:10.18653/v1/n18-1202.
- [4] J. Howard, S. Ruder, Universal language model fine-tuning for text classification, volume 1, 2018. doi:10.18653/v1/p18-1031.
- [5] S. Merity, C. Xiong, J. Bradbury, R. Socher, Pointer sentinel mixture models, 2017.
- [6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Łukasz Kaiser, I. Polosukhin, Attention is all you need, volume 2017-December, 2017.
- [7] J. Devlin, M. W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, volume 1, 2019.
- [8] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, Improving language understanding with unsupervised learning (2018).
- [9] Express.js, Express - node.js web application framework, n.d. URL: <https://expressjs.com/>, accessed in 10-04-2021.
- [10] Node.js, Node.js v16.4.0 documentation, n.d. URL: https://nodejs.org/api/child_process.html, accessed in 17-03-2021.
- [11] L. F. Cunha, J. C. Ramalho, Ner@di, 2021. URL: <http://ner.epl.di.uminho.pt/>, accessed in 09-10-2021.
- [12] F. Souza, R. Nogueira, R. Lotufo, BERTimbau: pretrained BERT models for Brazilian Portuguese, in: 9th Brazilian Conference on Intelligent Systems, BRACIS, Rio Grande do Sul, Brazil, October 20-23 (to appear), 2020.
- [13] J. A. W. Filho, R. Wilkens, M. Idiart, A. Villavicencio, The brwac corpus: A new open resource for brazilian portuguese, 2019.
- [14] J. Devlin, M. Chang, K. Lee, K. Toutanova, BERT: pre-training of deep bidirectional transformers for language understanding, CoRR abs/1810.04805 (2018). URL: <http://arxiv.org/abs/1810.04805>. arXiv:1810.04805.
- [15] A. Alvi, P. Kharya, Using deepspeed and megatron to train megatron-turing nlg 530b, the world’s largest and most powerful generative language model, 2021. URL: <https://www.microsoft.com/en-us/research/blog/using-deepspeed-and-megatron-to-train-megatron-turing-nlg-530b-the-worlds-largest-and-most-powerful-generative-language-model/>, accessed in 15/10/2021.
- [16] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. 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. M. 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, D. Amodei,

Language models are few-shot learners, volume 2020-December, 2020.

- [17] HuggingFace, Summary of the tokenizers, 2021. URL: https://huggingface.co/transformers/tokenizer_summary.html, accessed in 23/08/2021.
- [18] R. Sennrich, B. Haddow, A. Birch, Neural machine translation of rare words with subword units, volume 3, 2016. doi:10.18653/v1/p16-1162.
- [19] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. Le Scao, S. Gugger, M. Drame, Q. Lhoest, A. Rush, Transformers: State-of-the-art natural language processing, in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, Association for Computational Linguistics, Online, 2020, pp. 38–45. URL: <https://aclanthology.org/2020.emnlp-demos.6>. doi:10.18653/v1/2020.emnlp-demos.6.

A. ARCANO Sequence Diagram.

