

Universität zu Köln
Institut für Digital Humanities

Schwerpunkt 1: Verarbeitung von Textdaten

Die digitale Gesellschaft

Prof. Dr. Nils Reuters

WiSe 2023-2024

Modulprüfung

Classification of *League of Legends* Characters using a Transformer Model

von

Pascale Boisvert

MA - Informationsverarbeitung

1. Fachsemester

Eifelstraße 2

50677 Köln

Tel. 0178/2857148

pboisver@smail.uni-koeln.de

Matrikelnummer:

7349418

Table of contents

Table of contents.....	1
Introduction.....	2
State of Research.....	2
What are Transformer Models?.....	3
Report on the Practical Part.....	4
Fine-Tuning a Model.....	4
Testing the Model.....	5
Observations.....	5
How can the Results be Improved?.....	6
Conclusion.....	6
Literature.....	8
Figures.....	10

Introduction

In the field of natural language processing, many models have been developed to perform and automatise many tasks. They also have been used to analyze large corpuses of texts and extract information from them. They can be trained to sort texts, based on different elements, such as general sentiments, conversations and categories, as long as there is a large enough dataset to train the model¹. Apart from the required large dataset, it also requires a lot of computer power and money to train such models in an efficient way². Luckily, there exist already so-called “pre-trained” models that have already been trained on a very large dataset, to achieve specific tasks, such as sentiment analysis or text categorization. They then only require to be fine-tuned on a more specialized dataset and task to be able to theoretically perform well, at much lower costs. Such models can be found on the *Hugging Face* platform for example, where a large collection of pre-trained models and datasets are stored and can be used by its user to learn, try and test different tasks that do not only touch natural language processing³, but also image generation and classification as well as audio tasks. This then presents an interesting approach for the subject of this paper. The main goal is to try and fine-tune a model so that it can perform a classification task on a specific subject, the classification of *League of Legends* (Riot Games, 2009) characters according to their class, based on their biographies and short-stories. *League of Legends* is a multiplayer online battle arena (MOBA) developed back in 2009 by Riot Games. It consists of two teams of five players going face to face in order to defeat the opposing team. For this kind of game to work, certain roles have to be filled, such as a “tank” that can take a lot of damages while disrupting the opposing players, a “damage dealer” often referred to as “dps” that outputs a lot of damages, a “support” that can heal and boost the team and so on. Depending on the game, these roles may vary a bit. In *League of Legends*, a character can be classified as one of the following six classes: “assassin”, “fighter”, “mage”, “marksman”, “support” or “tank”. Each of these classes have abilities that can help them succeed in their respective role, such as a lot of health points (HPs), high mobility or high damage outputs. However, what makes *League of Legends* a bit more interesting in the classification of their role, is the fact that each character possesses a biography, a back-story, a setting and complementary information about its life, the so called “lore” that can be found in an external website: <https://universe.leagueoflegends.com>. The characters have very diverse and complete stories attached to them, so it was interesting to see if any pattern could be detected from their biographies and stories to see if it was enough, or specific enough, information to accurately categorize them in their right class. As someone who enjoys reading these stories and learning about the storyworld surrounding the game and its characters, it was a great opportunity to combine the two fields together. It would also permit us to see how well a model can be fine-tuned using a limited dataset to train and test. The game possesses only 168 characters and even when combining the biographies and short-stories the dataset totals only about 336 entries that then need to make up the train and test datasets. It also allowed us to see how well can a model be fine-tuned into picking up small context clues that could point out to a character's class or even, if the stories of the character actually reflect much of their actual class in the game or if it is only flavor. The following parts will take a look at how the research on such subjects is, where is it headed and what resources are available, then it will take a look at what are transformers models, how the process went for this specific case, how the research was conducted and what were the results and to complete, it will take a look at how it can be expended and further researched.

State of Research

The research state, depending on which aspect is looked at, is either very well-developed with good resources available or very little explored with very few resources to go by. On the first element, concerning natural language processing and categorization, the *Hugging Face* platform offers many options and possibilities. They have a large selection of pre-trained models, tutorials, datasets and other helpful resources concerning the technical part of the project. They offer plenty of documentation on the possible tasks and how to go around using their platform⁴. However, when looking at the aspect of large language models in the gaming world, some aspects have been more researched than others. For example,

¹ „What Is a Large Language Model (LLM)“. 2023. GeeksforGeeks. June 4. <https://www.geeksforgeeks.org/large-language-model-llm/>.

² „How do Transformers work? - Hugging Face NLP Course“. 2024. Visited on February 8. <https://huggingface.co/learn/nlp-course/en/chapter1/4>.

³ „Transformers, what can they do? - Hugging Face NLP Course“. 2024. Visited on March 3. <https://huggingface.co/learn/nlp-course/en/chapter1/3>.

⁴ „Hugging Face - Documentation“. 2024. Visited on March 10. <https://huggingface.co/docs>.

there is a good amount of papers using LLMs in game development, be it in character design, level design or even as coding help, this research here tried, and successfully, implemented a game of table tennis where the player could change the paddle and ball and the game would give out what was the most possible reaction. It also had to swap the two elements according to what the player said⁵. It is also explored how these LLMs could function as evolutive conversing agents with the player, while the player plays⁶. For the field of categorization concerning the gaming world, it is often the players' chats and/or actions that are observed, as a way to evaluate the toxicity and other communications techniques used in such games⁷. For MOBA-genre artificial intelligence has been used for helping game analytics as the following studie shows by mapping the researches done on the subject⁸. However, research concerning the classification of the characters based on their lore was not found. One would find more success by going into the literature side and even there, character classification is not a widely spread subject, at least not in the english languages. One paper that is similar in certain ways, was this "*Detecting Spells in Fantasy Literature with a Transformer Based Artificial Intelligence*" from Moravek M., Müller A., and Zender A.⁹ They also used a transformer model to fine-tune it on a relatively small dataset and evaluate its capacity to achieve context-based phrase recognition. The main subject here being *Harry Potter* (J.K. Rowling, 1997-2007) and the recognition of spells in fantasy settings. The similarities lie in the smaller available dataset and the use of the transformer model, however, the main task stays different even with similar problematics. Concerning the resources for the *League of Legends* aspect, the datasets that exists are ones from various players' chat logs, but for the lore, the biographies and shot-stories, everything had to be done, as they were, to the best of my knowledge, only the texts available on the website.

What are Transformer Models?

It is first important to understand what transformer models are. They are language models that have been trained, self-supervised, on large text corpuses in order for them to gain a statistical understanding of human language. They however require further supervised training in order to accomplish specific tasks, this is the process called "fine-tuning" and is meant to achieve transfer learning. The *Hugging Face* platform possesses many pre-trained models and allows users to fine-tune and share these models, but also to upload their own pre-trained models. This has the objective to greatly reduce the environmental impact of model training as well as hosting models that are trained on an always growing corpus¹⁰.

These models can easily be used by anyone and that is because of the `pipeline()` function. This function connects the models to the default pre-and post-processing steps required for them to function¹¹. The first step, pre-processing, is where the input text is being converted into numbers to allow the models to understand them. The texts are being tokenized, separated into smaller parts called tokens, using a `tokenizer()` method, usually the `AutoTokenizer()` as it fetches the default data associated with the selected model¹². This method outputs a dictionary that can then be converted into tensor, here using *Pytorch*. It is also simple to get the right architecture, the required information the model needs to perform the wanted task and output it in a way a human could understand it. This information is easily fetched

⁵ Roberts, Jasmine, Andrzej Banburski-Fahey, und Jaron Lanier. 2022. „Steps towards prompt-based creation of virtual worlds“. arXiv. <http://arxiv.org/abs/2211.05875>.

⁶ Chalamalasetti, Kranti, Jana Götze, Sherzod Hakimov, Brielen Madureira, Philipp Sadler, und David Schlangen. 2023. „Clembench: Using Game Play to Evaluate Chat-Optimized Language Models as Conversational Agents“. arXiv. <http://arxiv.org/abs/2305.13455>.

⁷ Murnion, Shane, William J. Buchanan, Adrian Smales, und Gordon Russell. 2018. „Machine learning and semantic analysis of in-game chat for cyberbullying“. *Computers & Security* 76 (July): 197–213. doi:10.1016/j.cose.2018.02.016

⁸ Costa, Lincoln Magalhães, Anders Drachen, Francisco Carlos Monteiro Souza, und Geraldo Xexéo. 2024. „Artificial Intelligence in MOBA Games: A Multivocal Literature Mapping“. *IEEE Transactions on Games*, 1–23. doi:10.1109/TG.2023.3282157.

⁹ Moravek, Marcel, Alexander Zender, und Andreas Müller. 2023. „Detecting Spells in Fantasy Literature with a Transformer Based Artificial Intelligence“. arXiv. <http://arxiv.org/abs/2308.03660>.

¹⁰ „How do Transformers work? - Hugging Face NLP Course“. 2024. Visited on February 8. <https://huggingface.co/learn/nlp-course/en/chapter1/4>.

¹¹ „Transformers, what can they do? - Hugging Face NLP Course“. 2024. Visited on March 3. <https://huggingface.co/learn/nlp-course/en/chapter1/3>.

¹² „Tokenizers - Hugging Face NLP Course“. 2024. Visited on March 12. <https://huggingface.co/learn/nlp-course/en/chapter2/4>.

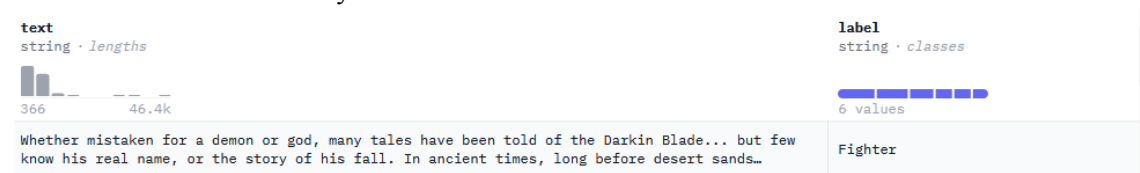
using the needed class, in this case the `ForSequenceClassification` class¹³. In the post-processing step, the models' outputs logits are being converted into their respective associated labels using `id2label`.

Report on the Practical Part

The goal of the project is to see how well the fine-tuned model can categorize the different *League of Legends* characters based on their biographies and short-stories. For this to work, a model has to be fine-tuned with the dataset of the characters and then tested with the "test" part of the dataset. It also has, as a goal, to see how the language of the dataset can influence the accuracy score of the model, for this, the model was fine-tuned in four different instances. I also wanted to explore how much the number of epochs had on the capacity of the model, without having it go into an overfitting behavior, but also evaluate how the size of the dataset can affect said results. This is why, in the end, there were four different fine-tuned models and four different datasets: one with the biographies in English, another with the biographies in French, another with the German biographies and the last one was a dataset combining the English biographies and short-stories, as a try to get a bigger train and test set. Each of these fine-tuned models were then tested on their respective "test" splits and their accuracy calculated.

Fine-Tuning a Model

To be able to achieve the project, there were a few steps that needed to be done. The first one was to get to know the *Hugging Face* platform and understand how to use their transformer models and fine-tune it to a specific dataset. Luckily, they offer a variety of tutorials to manage the simplest tasks and they also have a community forum where one can refer to in case of errors or problems¹⁴. The next step was to figure out which task would be best suited for the need of the classification task. As there are six possible labels: "assassin", "fighter", "mage", "marksman", "support" or "tank", the zero-shot classification seemed like the best choice. This kind of classification also allowed me to not rely on the labels the pre-trained model received, which was an important aspect as this classification can be considered very niche¹⁵. Then, it was only a question of filtering the pre-trained models according to the wanted tasks and choosing one from the filtered options. In this case, the "facebook/bart-large-mnli" was the most popular one available¹⁶. Then, all there was left to do was to fine-tune the chosen model, by training it on a specific dataset. So, the dataset had to be done. I had no experience in creating or even using a dataset with the *Hugging Face* platform, so I first had to explore the other sets in order to decide how to make the one that I was going to be using. It ended up being a simple .csv file, with two columns: one where the text would be stored and one where the label associated is specified, as shown in the Figure 1. It can also be found in the datasets library¹⁷.



text	label
string · lengths 366 — 46.4k	string · classes 6 values
Whether mistaken for a demon or god, many tales have been told of the Darkin Blade... but few know his real name, or the story of his fall. In ancient times, long before desert sands...	Fighter

Figure 1: Example of the dataset once imported in the Hugging Face hub

The dataset, even if small, was separated in a typical "split" manner, where 80% of the set will be used for training and the other 20% will be used for testing the model once it is trained. The dataset also had to be done manually, as the website does not contain an API that one could take or use and this took quite a while in the process. It was also important to get the right label for each character, as they were indicated on their page. This process was done four times, one for the biographies in three languages, English, French and German and then for the short-stories, but only in English. This was not done all at the same

¹³ „Behind the pipeline - Hugging Face NLP Course“. 2024. Visited on February 20.
<https://huggingface.co/learn/nlp-course/en/chapter2/2>.

¹⁴ „Hugging Face Forums“. 2024. Hugging Face Forums. Visited on March 20.
<https://discuss.huggingface.co/>.

¹⁵ „Transformers, what can they do? - Hugging Face NLP Course“. 2024. Visited on March 3.
<https://huggingface.co/learn/nlp-course/en/chapter1/3>.

¹⁶ „facebook/bart-large-mnli · Hugging Face“. 2024. January 4.
<https://huggingface.co/facebook/bart-large-mnli>.

¹⁷ „Flamgrise/Biographies · Datasets at Hugging Face“. 2024. Visited on March 31.
<https://huggingface.co/datasets/Flamgrise/Biographies>.

time, the different language datasets came later, as I decided I also wanted to see if the language had an impact on the fine-tuning and results of a model.

The code for the fine-tuning was done in a *Google Colab* environment¹⁸, to mostly help with the resources, as training can take quite a lot of power. I followed a text classification tutorial from *Hugging Face* and adapted a few aspects of it after many different error codes that came up when trying to test it¹⁹. One of the biggest problems was the nesting of strings, which would not allow for the trainer to properly be used. To solve this, I had to take a few different steps. The first one was to delete both the “text” and “label” columns of the dataset once it was tokenized. The second step was to make sure that the labels were not nested, to then be able to assign them an integer, according to the previously done mapping of the labels. This step can be seen in the code attached to the `tokenize()` function, or see Figure 2.

Another important aspect for me was to calculate the accuracy score of the model during its training. I wanted to see how it would fair and could also monitor how the score would change according to the number of epochs that was specified in the training arguments. This came to a few problems as the function would simply not work. After looking into it, it was suggested to calculate the F1-score instead of the accuracy, to go over the problem, which is what was done (Figure 3).

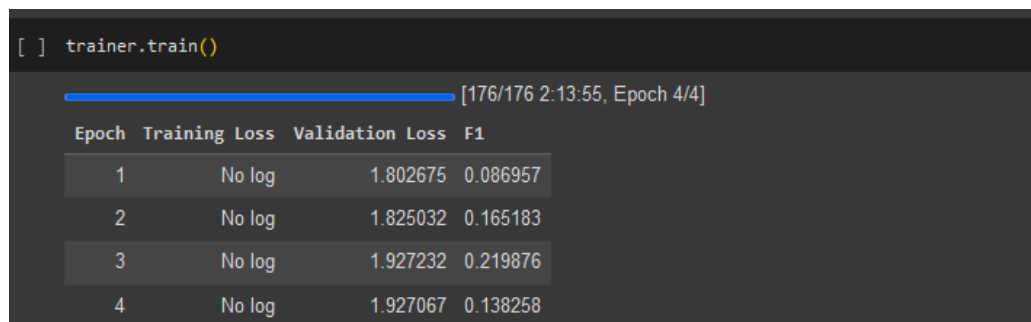
Once most of these problems were solved, I could finally start the training loop, only to be stopped about half an hour later with a “Target out of bound” error (Figure 4). This was solved with the help of a parameter in the `AutoModelForSequenceClassification` method that needed to be set to `True`. (Figure 5) After this correction, the training-loop could be started and it was finally working. However, I had to play a lot with the parameters around as *Google Colab* limits the RAM that can be used and training with batches any bigger than three would make the notebook crash and the training had to be restarted. Then, even only a small amount of epochs took quite some time to get trained, needing anywhere between three to sixteen hours. This greatly reduced my capacity of trying out playing with the parameters, but I decided that the optimal number of epochs would be 12 for this task and what I could have in terms of power. I tried upping the epoch all the way to 25, but when the notebook disconnected after 18 hours of training and it not being finished, and also seeing that the results were not getting better, I left the number of epoch much lower (Figure 6). The whole code used for training can be found as .py files in the zip-folder in the train sub-folder.

Testing the Model

Once the training loop was done, the model could be saved and imported into my *Hugging Face* hub, as seen on figure 7. Using the *Hugging Face* transformer library, it was quite easy to get the wanted model and use it on the selected dataset. Then, calculating the accuracy could be done with a simple function. These codes can be found in the test sub-folder.

Observations

There are a few observations that can be made from looking at the different results after training and testing the datasets. The first one is, the number of epochs on which the dataset was trained had little influence on the overall accuracy, as shown by figures 8 and 9 which concerns a fine-tuned model on only english biographies and the accuracy resulting from the testing of said model.



Epoch	Training Loss	Validation Loss	F1
1	No log	1.802675	0.086957
2	No log	1.825032	0.165183
3	No log	1.927232	0.219876
4	No log	1.927067	0.138258

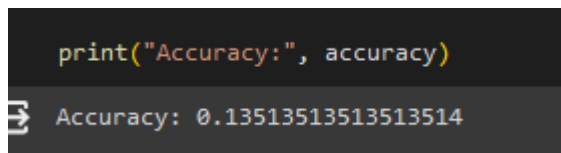
Figure 8: F1-Score of the English-Biographies Fine-Tuning model

¹⁸ „Google Colaboratory“. 2024. Visited on March 25.

https://colab.research.google.com/drive/1QsiM6u1KSms8l_tYVvKOkI8JjfHjCdH84?usp=sharing.

¹⁹ „Text classification“. 2024. Visited on March 10.

https://huggingface.co/docs/transformers/en/tasks/sequence_classification.



```
print("Accuracy:", accuracy)
Accuracy: 0.13513513513513514
```

Figure 9: Accuracy of the model being tested on the test dataset

The same can be observed when looking at the german biographies results in figure 10 and 11 and the french biographies in figure 12 and 13. It is also interesting to observe that the model that was fine-tuned on the French dataset shows a higher accuracy when being tested even if its original F1-scores are much lower. It could be due to luck or maybe the fact that the fine-tuning worked particularly well in the French language. However, the problem, as said before, with such little datasets, it is hard to see any real influence on the results, but it can be said that the scores are less than acceptable for the tests to be said to be a success²⁰. The other try I made was combining both the short-stories and the biographies of the characters to double the size of the dataset in the hope it would help the results. After a first training crash and the decision to reduce the number of epochs the results were worse than expected: higher F1-score when training but also the lowest accuracy score when being tested, as demonstrated by figures 14 and 15.

From the above results, it is tempting to say that the “facebook/bart-large-mnli-model” can be fine-tuned in different languages and similar results can be obtained, at least in English, German and French. It is also possible to observe that the number of epochs only slightly changes the results and that even in one case, the accuracy was much more worse than the F1-Scores that it could have been going into an overfitting context, where the model simply learned which label belong to which text but could not do it on other datasets. It is however important to note that the datasets remained quite small and that fine-tuning, for such a specific task on such a small dataset can lead to not having satisfactory results. It is also possible that the biographies and short-stories of the character simply do not represent their classes in game and that no subtleties or context clues can be found to accurately assign them their right class.

How can the Results be Improved?

The results of the research are quite low. A good F1-score and accuracy would ideally find itself to be in the higher 0.8-0.9. There would probably be ways to better the obtained results, even though the dataset stays small. A few options, linked with the tokenization of the text, would be worth looking into, as improved results have been shown in other studies²¹. One would be to adjust how the texts are split, either by sentences, paragraphs or to the maximum allowed by the model, which would be called a sequence split. This would reduce the context-length, when using sentence or paragraph split or would make it larger, when using sequence split. Comparing the results could be interesting to see if the context-length plays an important role in the case of character classification. Another route could be to analyze the results using the Transformer-Interpret library to identify which tokens weigh the most in the classification and find solutions from there. However, I was not aware of the existence of the library when doing the testing and as it takes many hours to go through a cycle, time started running out.

Another option would be to adjust the tokenizer, first by making sure that the fantasy words or fictive places are recognized as one entity to help with the weighting or even by displaying the different weight of the tokens, and observe what words make the models go in one direction or another. This would ask again a lot more time, but could be done and might help better the parameters when fine-tuning a model, hopefully achieving better results. Also, maybe using a different pre-trained model might help the results in some ways.

Conclusion

To summarize, language models most likely have the capacity to categorize the *League of Legends* characters to their in-game class based on their biographies and short-stories, if these stories fit the class’s archetype. However, as it currently stands the stories differ vastly and there are too few available characters in order to produce a consistent and big dataset that would permit an efficient fine-tuning on an already pre-trained model. This lack of quantity is a big factor at play, especially since the nature of

²⁰ „F1 Score in Machine Learning“. 2023. GeeksforGeeks. Dezember 27. <https://www.geeksforgeeks.org/f1-score-in-machine-learning/>.

²¹ Moravek, Marcel, Alexander Zender, und Andreas Müller. 2023. „Detecting Spells in Fantasy Literature with a Transformer Based Artificial Intelligence“. arXiv. <http://arxiv.org/abs/2308.03660>.

story-writing differs greatly between authors as well as personal stories of the characters. It is important to remember that when working with smaller datasets to fine-tune a transformer model, one should put a particular attention when tokenizing the data in order to help the model as much as possible as well as looking into the optimized training-parameters that are usually given when reading about the model. It can also be a good idea to keep in mind if it is possible to find similarities in other subjects or themes to try and furnish the dataset a bit more, again to heighten the chances of a successful fine-tuning.

On a personal aspect, I learned a lot about fine-tuning a model and how important each step can be in the process. It was also an opportunity to deepen my knowledge on the programming aspect of fine-tuning of a model as well as learning about the transformer models. They can be a great way to reduce the environmental footprint of training different models to do specific tasks as well as help to open the research to a bigger audience. I also learned that the semantic used in text can cause great confusion when trying to apply it to a different context, especially if there is a clear lack of repetition in the training set, but it is however not impossible to mix different universes to achieve a better result as models can then find similarities somewhat easier.

As next, in link with this specific research it can be interesting to enlarge the dataset, either by waiting for more characters to come out or by mixing characters from another game, for example *Dota 2* (Valve, 2013) if one can find short biographies. It can also be interesting to try and mix the different languages together and try to achieve a higher accuracy score, but it seems that multi-language models usually perform worse than unilingual ones²². Improving the tokenization of the texts could lead to better results, helping the model better understand and linking different literary aspects to these specific in-game classes. Improving the training parameters could also most likely help, either using a better performing machine or giving it more time on a machine that would have less chances of crashing, for example, not using internet-linked *Google Colab*, where internet might cut at any moment, but running the program locally on the PC. These steps will probably help achieve better results in the model and from there allow people to use the model in a reliable way, when using it to categorize the *League of Legends* characters based on their biographies and short-stories. From there, the model can then be extended to other languages and even tried on different games with different characters and similar classes to see if it can be used on multiple MOBAs, but also be extended to different game-genres.

²² „Hugging Face - Documentation“. 2024. Visited on March 10. <https://huggingface.co/docs>.

Literature

1. „Behind the pipeline - Hugging Face NLP Course“. 2024. Visited on February 20. <https://huggingface.co/learn/nlp-course/en/chapter2/2>.
2. Chalamalasetti, Kranti, Jana Götze, Sherzod Hakimov, Brielen Madureira, Philipp Sadler, und David Schlangen. 2023. „Clembench: Using Game Play to Evaluate Chat-Optimized Language Models as Conversational Agents“. arXiv. <http://arxiv.org/abs/2305.13455>.
3. Costa, Lincoln Magalhães, Anders Drachen, Francisco Carlos Monteiro Souza, und Geraldo Xexéo. 2024. „Artificial Intelligence in MOBA Games: A Multivocal Literature Mapping“. IEEE Transactions on Games, 1–23. doi:10.1109/TG.2023.3282157.
4. „F1 Score in Machine Learning“. 2023. GeeksforGeeks. December 27. <https://www.geeksforgeeks.org/f1-score-in-machine-learning/>.
5. „facebook/bart-large-mnli · Hugging Face“. 2024. January 4. <https://huggingface.co/facebook/bart-large-mnli>.
6. „Flamgrise/Biographies · Datasets at Hugging Face“. 2024. Visited on March 31. <https://huggingface.co/datasets/Flamgrise/Biographies>.
7. „Google Colaboratory“. 2024. Visited on March 25. https://colab.research.google.com/drive/1QsiM6u1KSms8l_tYVvOkI8JfHjCdH84?usp=sharing.
8. „How do Transformers work? - Hugging Face NLP Course“. 2024. Visited on February 8. <https://huggingface.co/learn/nlp-course/en/chapter1/4>.
9. „Hugging Face - Documentation“. 2024. Visited on March 10. <https://huggingface.co/docs>.
10. „Hugging Face Forums“. 2024. Hugging Face Forums. Visited on March 20. <https://discuss.huggingface.co/>.
11. Moravek, Marcel, Alexander Zender, und Andreas Müller. 2023. „Detecting Spells in Fantasy Literature with a Transformer Based Artificial Intelligence“. arXiv. <http://arxiv.org/abs/2308.03660>.
12. Murnion, Shane, William J. Buchanan, Adrian Smales, und Gordon Russell. 2018. „Machine learning and semantic analysis of in-game chat for cyberbullying“. Computers & Security 76 (July): 197–213. doi:10.1016/j.cose.2018.02.016.
13. Roberts, Jasmine, Andrzej Banburski-Fahey, und Jaron Lanier. 2022. „Steps towards prompt-based creation of virtual worlds“. arXiv. <http://arxiv.org/abs/2211.05875>.
14. „Text classification“. 2024. Visited on March 10. https://huggingface.co/docs/transformers/en/tasks/sequence_classification.
15. „Tokenizers - Hugging Face NLP Course“. 2024. Visited on March 12. <https://huggingface.co/learn/nlp-course/en/chapter2/4>.
16. „Transformers, what can they do? - Hugging Face NLP Course“. 2024. Visited on March 3. <https://huggingface.co/learn/nlp-course/en/chapter1/3>.
17. „What Is a Large Language Model (LLM)“. 2023. GeeksforGeeks. June 4. <https://www.geeksforgeeks.org/large-language-model-llm/>.

18. Wu, Bin, Qiang Fu, Jing Liang, Peng Qu, Xiaoqian Li, Liang Wang, Wei Liu, Wei Yang, und Yongsheng Liu. 2018. „Hierarchical Macro Strategy Model for MOBA Game AI“. arXiv. <http://arxiv.org/abs/1812.07887>.

Figures

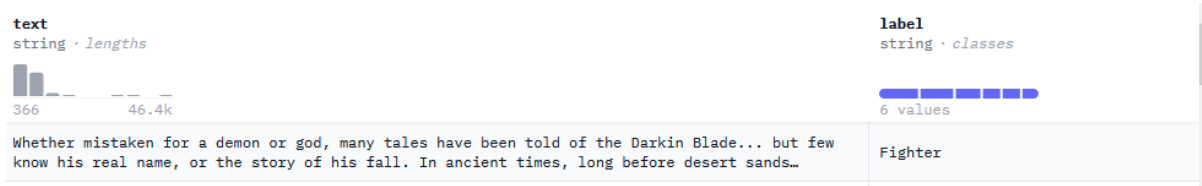


Figure 1: Example of the dataset once imported in the Hugging Face hub

```
import torch
label_mapping = {"Fighter": 0, "Mage": 1, "Assassin": 2, "Marksman": 3, "Tank": 4, "Support": 5}

def preprocess_function(batch_data):
    # Check if "text" key exists in batch_data
    if "text" not in batch_data:
        raise KeyError("Key 'text' not found in batch_data.")

    # Tokenize text with truncation and padding
    tokenized_text = tokenizer(batch_data["text"], truncation=True, padding='max_length', return_tensors="pt")
    tokenized_text = {key: tensor.squeeze() for key, tensor in tokenized_text.items()} # Remove the batch dimension

    # Remove the "text" column
    del batch_data["text"]

    # Handle labels
    if "label" in batch_data:
        labels = batch_data["label"]

        if isinstance(labels, list): # Check if labels are nested
            labels = [label_mapping[label.strip()] for label in labels] # Convert labels to integers
        else:
            labels = label_mapping[labels.strip()] # Convert single label to integer

        # Remove the "label" column
        del batch_data["label"]

        tokenized_text["labels"] = torch.tensor(labels, dtype=torch.long) # Convert labels to tensor of type long

    return tokenized_text

# Tokenize the data
tokenized_dataset = dataset.map(preprocess_function, batched=True)
```

Map: 100% 72/72 [00:00<00:00, 457.39 examples/s]

Figure 2: Pre-processing & Tokenization of the dataset

```
[ ] from transformers import EvalPrediction
import numpy as np
from sklearn.metrics import f1_score

def compute_metrics(pred):
    labels = pred.label_ids
    logits = pred.predictions[0] if isinstance(pred.predictions, tuple) else pred.predictions

    # Convert logits to PyTorch tensor and get predicted labels
    logits_tensor = torch.from_numpy(logits)
    predictions = logits_tensor.argmax(dim=-1).detach().cpu().numpy()

    # Calculate macro-averaged F1 score
    f1 = f1_score(labels, predictions, average='macro')

    result = {}
    result["f1"] = f1
    return result
```

Figure 3: Function used to calculate F1-Score during the training loop

```
IndexError: Target 4 is out of bounds.
```

Figure 4: Target is out of bound

```
from transformers import AutoModelForSequenceClassification, TrainingArguments, Trainer

model = AutoModelForSequenceClassification.from_pretrained('facebook/bart-large-mnli', ignore_mismatched_sizes=True, id2label=id2label, label2id=label2id)
```

Figure 5: Parameter that solved the Out of bound error

[1481/2200 18:04:06 < 8:47:01, 0.02 it/s, Epoch 16.82/25]

Epoch	Training Loss	Validation Loss	F1
1	No log	1.754505	0.077778
2	No log	1.794048	0.047619
3	No log	1.737199	0.082060
4	No log	1.736376	0.075269
5	No log	1.758021	0.048193
6	1.819600	1.765726	0.057471
7	1.819600	1.727531	0.075269
8	1.819600	1.725761	0.075269
9	1.819600	1.728198	0.075269
10	1.819600	1.719414	0.075269
11	1.819600	1.720702	0.075269
12	1.774200	1.724955	0.075269
13	1.774200	1.732788	0.075269
14	1.774200	1.727835	0.057471
15	1.774200	1.719785	0.075269
16	1.774200	1.726781	0.057471

Figure 6: F1-Scores of the trained model after crashing on epoch 16 from 25

```
[ ] from transformers import PushToHubCallback
# Save the trained model
trainer.save_model("ENG-full-fine-tuned")

# Push the model to the Hugging Face Hub
trainer.push_to_hub(commit_message="Fine-tuned BART model for League of Legends biographies classification")

# If you want to save the tokenizer and config files as well:
tokenizer.save_pretrained("ENG-full-fine-tuned")
model.config.save_pretrained("ENG-full-fine-tuned")
```

Figure 7: Code to save and import the model in Hugging Face

[] trainer.train()

[176/176 2:13:55, Epoch 4/4]

Epoch	Training Loss	Validation Loss	F1
1	No log	1.802675	0.086957
2	No log	1.825032	0.165183
3	No log	1.927232	0.219876
4	No log	1.927067	0.138258

Figure 8: F1-Score of the English-Biographies Fine-Tuning model

```
print("Accuracy:", accuracy)

Accuracy: 0.13513513513513514
```

Figure 9: Accuracy of the model being tested on the test dataset

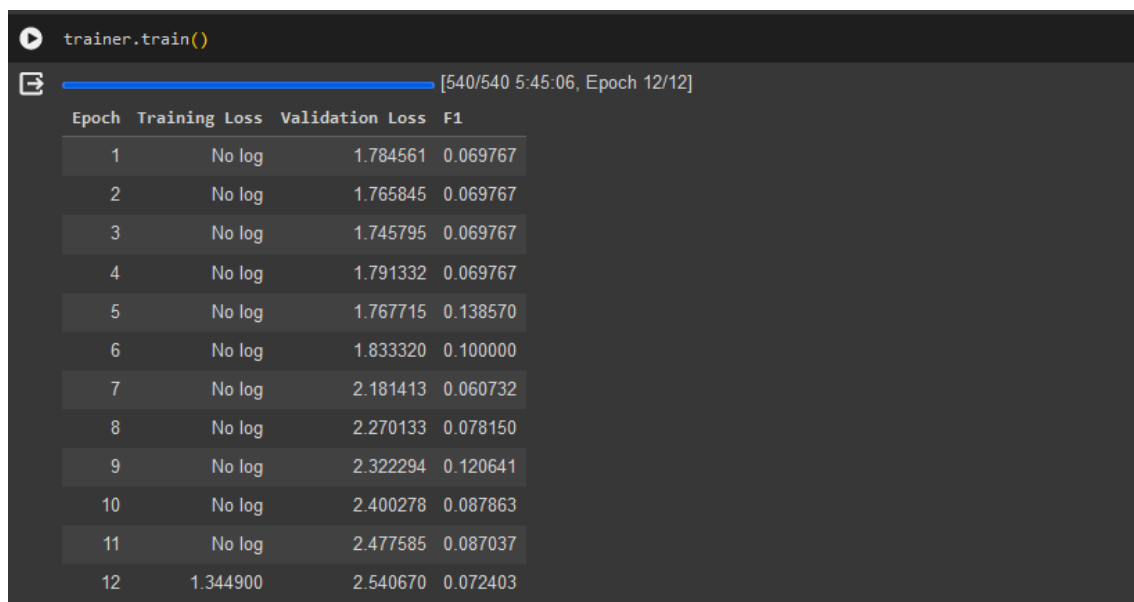


Figure 10: German Fine-Tuning, F1-Scores

```
print("Accuracy:", accuracy)
```

Accuracy: 0.14705882352941177

Figure 11: German tested accuracy of the model

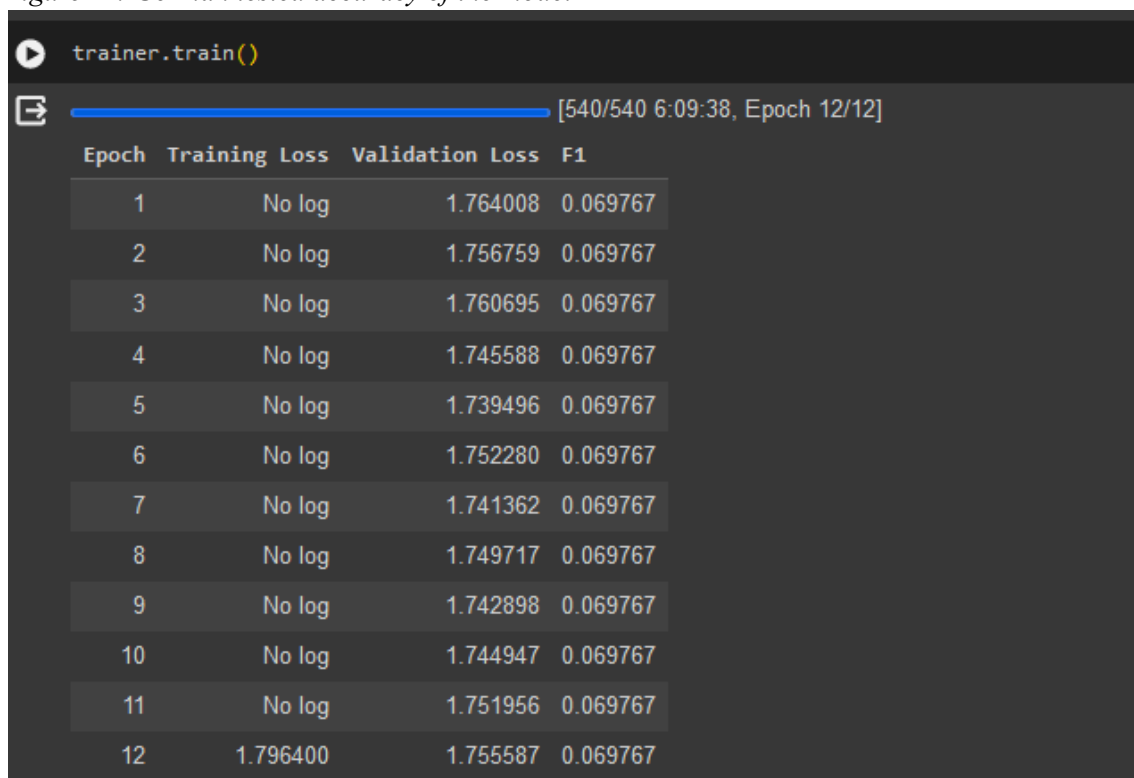


Figure 12: French Fine-Tuning, F1-Scores

```
print("Accuracy:", accuracy)
```

Accuracy: 0.23529411764705882

Figure 13: Tested accuracy for the model tested on French-Dataset

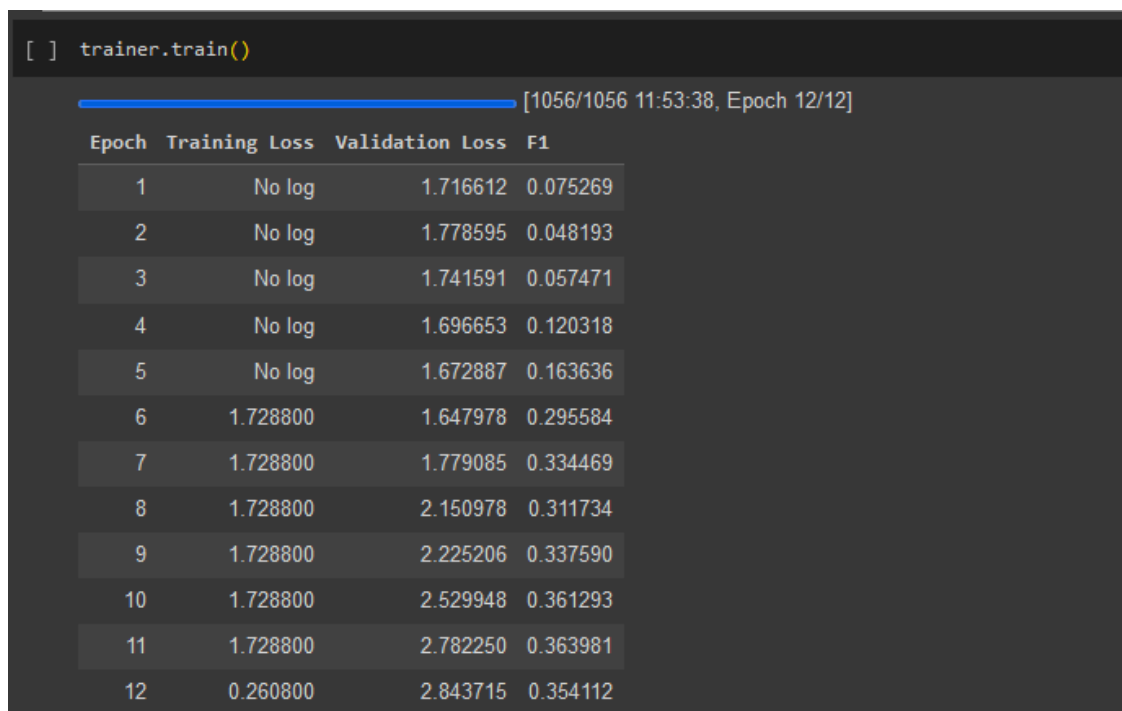


Figure 14: F1-Scores of the combined English datasets

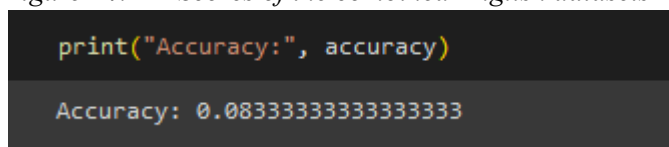


Figure 15: Accuracy of the model when tested

Hiermit versichere ich an Eides Statt, dass ich diese Bachelorarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Die Stellen meiner Arbeit, die dem Wortlaut oder dem Sinn nach anderen Werken und Quellen, einschließlich der Quellen aus dem Internet, entnommen sind, habe ich in jedem Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht. Dasselbe gilt sinngemäß für Tabellen, Karten und Abbildungen.
Diese Arbeit habe ich in gleicher oder ähnlicher Form oder auszugsweise nicht im Rahmen einer anderen Prüfung eingereicht.

Köln, 01.04.2024 _____ Unterschrift: Pascale Boimert