Universität Trier

Literary Metaphor Detection with LLM Fine-Tuning and Few-Shot Learning

Master Digital Humanities

12401603 Vertiefung Digital Humanities

Large Language Models in the Digital Humanities

Wintersemester 2023/2024

Univ.-Prof. Dr. Christof Schöch

Marina Spielberg

Matrikelnummer:1656491

Abgabedatum: 13.06.2024

Erklärung zur Hausarbeit

Hiermit erkläre ich, dass ich das Portfolio selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die aus fremden Quellen direkt oder indirekt übernommenen Gedanken als solche kenntlich gemacht habe.
Die Arbeit habe ich bisher keinem anderen Prüfungsamt in gleicher oder vergleichbarer Form vorgelegt. Sie wurde bisher nicht veröffentlicht.

Datum: 13.06.2024 Unterschrift: 

Table of Contents

List of Abbreviations i
1. Introduction 1
2. Related Work 2
    2.1 Metaphor Detection in NLP 2
    2.2 Metaphor Detection in the Digital Humanities 3
3. Methodology 5
    3.1 Task Description 5
    3.2 Transformers Framework: Fine-tuning DistilBERT 6
    3.3 SetFit Framework: Few-shot fine-tuning with all-MiniLM-L6-v2 7
4. Experimental Setup 8
    4.1 Dataset Management: Preprocessing and Data Analysis 8
    4.2 Transformers Implementation 11
    4.3 SetFit Implementation 12
5. Experiment Results 13
6. Discussion 15
7. Ethical Concerns 17
8. Conclusion 18
9. Works Cited 19
Appendix 23
    Appendix A: Overview of fine-tuning results for Transformers and SetFit 23
    Appendix B: Overview of fine-tuning emissions 24

List of Abbreviations

General abbreviations

Abbreviation	Definition
BERT	Bidirectional Encoder Representations from Transformers
DH	Digital Humanities
LLM	Large Language Model
MD	Metaphor Detection
NLP	Natural Language Processing
POS	Part-of-speech

Abbreviations for Jupyter notebooks referenced in this paper

Abbreviation	Name of file
EV	Evaluation_visualisation.ipynb
PA	Preprocessing_analysis.ipynb
ST	SetFit_training.ipynb
TT	Transformers_training.ipynb

1. Introduction[1]

The study of literary metaphors plays an integral part in literature-focused disciplines within the humanities. Computational metaphor detection (MD) has produced a wealth of approaches in the field of Natural Language Processing (NLP), where the focus lies on detecting everyday metaphors (Ptiček and Dobša). In the Digital Humanities (DH) literary MD has not been extensively studied yet. In their Graph Project, Kesarwani et al. and Tanasescu et al. have applied rule-based and statistical machine learning approaches on an English poetry corpus.

The aim of this paper is to further the field of literary MD by using the NLP state-of-the-art approach of fine-tuning Large Language Models (LLM) on four datasets from the Graph project. The goal is to assess in how far this method will improve classification performance. Considering the small size of the datasets, the few-shot learning approach SetFit is used alongside the traditional fine-tuning approach. Specifically, this paper wants to test two assumptions:

1. Fine-tuning the Transformer LLM DistilBERT and the Sentence Transformer LLM all-MiniLM-L6-v2 on the Grapheme project's datasets will yield better results than using statistical and machine learning approaches.
2. The few-shot learning approach SetFit applied to the model all-MiniLM-L6-v2 will outperform the fine-tuning method with DistilBERT.

The paper begins with an overview of current trends in MD within NLP and DH. The methodology chapter defines MD as a sentence-level classification problem and outlines the architecture, workflow and advantages of the Transformers and SetFit frameworks for the purposes of this paper. Then the experimental setup is described in more detail, which includes the data preprocessing process and data analysis as well as the Transformers and SetFit implementations. Emphasis is placed on coding decisions that deviate from the proposed guidelines of the frameworks. The results are then reported and discussed. Finally, the ethical concerns chapter draws attention to issues like research replicability, licensing, and computational resource usage. The deployment and application of the models fall outside of the scope of this paper.

[1] This paper uses the MLA citation style and the in-text referencing system. Code from the Jupyter notebooks is referenced in the format notebook name cell number : line number. The code created for this paper as well as the datasets and outputs are publicly available in the GitHub repository LLM_metaphor_detection. The resulting fine-tuned models can be found on Zenodo.

2. Related Work

To select an approach for literary metaphor detection it is beneficial to have an understanding of the theories, methods and datasets that have established themselves in this field. Since the bulk of research on this topic comes from NLP this chapter first introduces its most prevalent ideas before focusing on the current research in the Digital Humanities and how this paper can contribute to it.

2.1 Metaphor Detection in NLP

Metaphor Detection has been a topic of interest in NLP since 1975 and the research it produces is ongoing because "the task is not considered solved" (Dankin et al. 125). The theoretical basis for most of the MD research is the Conceptual Metaphor Theory by Lakoff and Johnson (Ptiček and Dobša 2, 24). This theory argues that "the essence of metaphor is understanding and experiencing one kind of thing in terms of another" (Lakoff and Johnson 5), that is, mapping a source domain like WAR to a target domain like ARGUMENT (Ptiček and Dobša 2). This cognitive mapping results in metaphorical linguistic expressions like "Your claims are *indefensible*" (ibid.). Since NLP is interested in studying natural language, it uses Conceptual Metaphor Theory to detect conversational metaphors, which are part of everyday language and novel metaphors, which are continually being coined (ibid. 2-4, 11).

Following the development of computational capabilities Ptiček and Dobša observe three stages of methodologies for MD: "Hand coded knowledge and use of lexical resources", "[s]tatistical and machine learning methods" and "[n]eural networks and word embeddings" (10). The first approach was used from 1975 to 2004 but was discontinued due to its complex and time-consuming nature in favour of the machine learning approach which was roughly adopted until 2016 (ibid. 10-12). This approach employed supervised learning algorithms for classification (e.g. support vector machine, logistic regression, random forest) and unsupervised learning like clustering for the MD task (ibid. 12-14). The current state-of-the-art method for MD involves utilizing neural networks and contextual word embeddings, which are synonymous with LLMs according to Ptiček and Dobša (2,15). Contextual word embeddings are based on the Transformer, a "sequence transduction model" (Vaswani et al. 10), whose distinctive features are the "self-attention layers" (ibid. 3) with an "encoder-decoder structure" (ibid. 2). This architecture enabled the vastly utilized procedure of fine-tuning Transformer-

based LLMs on downstream tasks like MD.[2] While fine-tuning is the most prominent method of using LLMs for the MD task in recent NLP research, 2024 saw the application of a new paradigm, prompt engineering, that uses zero-shot learning combined with task-specific formulated prompts, which does not require labelled data.[3]

To compare the MD evaluation results across different papers it became a convention to use the benchmark datasets VUA, TroFi and MOH in the NLP community (Ptiček and Dobša 9, 24). Comprising about 187 000 words, VUA (VU Amsterdam Metaphor Corpus) is the largest dataset and is therefore most frequently used (ibid. 7). TroFi (Trope Finder) and MOH are small datasets that contain 6435 and 647 sentences respectively (ibid. 8-9).

2.2 Metaphor Detection in the Digital Humanities

It is useful to examine research in the Digital Humanities to understand in how far employed theories, methods and datasets differ from NLP practises regarding the computational detection of literary metaphors. To my knowledge there are four papers that concern themselves with literal MD. Reinig and Rehbein proposed a machine learning method based on support vector machines to recognise metaphors in German expressionist poetry which they trained on a self-constructed dataset (149,153-155). Toker et al. used LLMs on their own Early Medieval Hebrew poetry dataset (1-5).

In their Graph Project, Kesarwani et al. and Tanasescu et al. developed models to detect poetic metaphors in English, which hitherto has neither been done in NLP nor Digital Humanities and published their results in the papers “Metaphor Detection in a Poetry Corpus” and “Metaphor Detection by Deep Learning and the Place of Poetic Metaphor in Digital Humanities”. Contrary to NLP’s concern with every-day metaphors these papers are interested in literary metaphors, which can be regarded as being the opposite of conventional metaphors (Kesarwani 1). Diverging from the Conceptual Metaphor Theory by Lakoff and Johnson, the authors based their research on Neuman’s observation of four specific part-of-speech (POS) sequences that often signify metaphorical phrases (Neuman 2-3). In particular, the authors focused on detecting type I metaphors, where a “subject noun is associated with an object noun via a form of the copula verb ‘to be’” (Neumann 2), that is, a Noun-Verb-Noun sequence.

[2] For an in-depth overview of research that used the discussed approaches from 1975 to 2023 see the review “Methods of Annotating and Identifying Metaphors in the Field of Natural Language Processing” by Ptiček and Dobša.

[3] See “Enhancing Metaphor Detection through Soft Labels and Target Word Prediction” by Jia and Li for detailed descriptions and results of using the prompt-based approach.

Kesarwani et al. added the sequence "Noun-Verb-Det-Noun" (2) to this concept. Thus, an example for the type I metaphor is the sentence "As if the *world were a taxi*" (ibid.).

Since no annotated poetry dataset for detecting metaphors has existed before, the authors created the PoFo (Poetry Foundation) dataset, comprising 680 sentences that include type I metaphors scraped from 12830 poems from the Poetry Foundation[4] (Kesarwani 2-3; "PoFo Corpus"). Considering that "[a]nnotating metaphor is not a trivial task" with "[b]orderline cases" and "ambiguity" (Kesarwani et al. 3) the meticulous PoFo annotation produced an inter-annotator-agreement of 72.94% and thus, can be considered good quality (ibid.). Its shortness is due to the fact that only type I metaphor tag sequences were included (ibid). In addition to PoFo, the authors used the aforementioned NLP benchmark datasets TroFi and MOH for training and testing. For better classification results the authors raised the sample size by creating a concatenated dataset of TroFi and MOH and 28% of PoFo for training and tested it on 72% of PoFo examples (ibid. 4-5). It is worthwhile to note that the sentences from TroFi and MOH, being sampled from the 1988-1989 Wall Street Journal Corpus and WordNet respectively, do not likely contain poetic metaphor (Birke and Sarkar 330, Mohammad et al. 2).

The deployed MD methods of the Graph Project mirror the progression of methods in the NLP tradition. Firstly, the authors experimented with rule-based (Abstract-Concrete and Concrete Category Overlap) and statistical (machine learning classifiers) approaches (Kesarwani 4-5). In the second paper, deep learning with convolutional neural networks was used (Tanasescu 123). However, contextual neural networks have not been tested on the datasets yet. The F1 scores for the rule-based and machine learning models were 0.669 for PoFo, 0.827 for TroFi, 0.779 for MOH and 0.781 for the aggregated dataset (Kesarwani 5-6). The deep learning method showed the best results with an F1 score of 0.822 on the aggregated dataset of 4870 sentences (Tanasescu 124). Due to the small sample sizes of the other datasets the convolutional neural network models yielded worse results than the machine learning models (Tanasescu 125). Reflecting on their research results the authors remarked that for the poetry MD task "non-poetry data are as helpful as poetry data" (Kesarwani 5; Tanasescu 124), referencing the better precision results of the combined dataset in comparison to the PoFo dataset (0.759 and 0.662).

To my knowledge there have been no further attempts to improve the results of the Graph Project's experiments. Therefore, this paper wants to use the same datasets and take their results as a baseline to test whether fine-tuning a LLM improves the performance on the MD

[4] See the Poetry Foundation website: www.poetryfoundation.org/.

task since this method proved to be most successful in recent NLP research. Additionally, it will address the issue of needing a "substantial amount of [poetic metaphor] data" (Tanasescu 122) for LLM training by utilizing few-shot learning which is suited for small datasets. The next chapter defines the task description of this paper and illustrates the chosen frameworks for fine-tuning and few-shot learning.

3. Methodology

This chapter introduces the methods of detecting literary metaphors by fine-tuning LLMs. First, it defines the MD task as a sentence-level classification problem. Then, it presents the chosen frameworks and models, Transformers with DistilBERT and SetFit with all-MiniLM-L6-v2, by explaining their workflow, properties, and advantages.

3.1 Task Description

There are various concepts of defining what part of a sentence should be considered for the MD task. Kesarwani et al. and Tanasescu et al. define the MD task on a POS-level, that is, they annotate a POS tag sequence in a sentence and let their model classify this sequence as metaphor or non-metaphor (Kesarwani 2). This annotation practice can be formalized as follows: If a sentence with k words is defined as "$s = \{w_1, w_2, ..., w_k\}$" (Ma et al. 34), the POS-level classification approach is characterized as predicting if the POS tag sequence w_i-$w_n \in s$ in a sentence s is metaphorical or literal. The TroFi and MOH datasets are annotated on the word-level (Birke and Sarkar 335-336; Mohammad et al. 25). This means that "[g]iven a sentence s and an aspect word $w_i \in s$ [the classifier should] predict the metaphoricity label l_i associated with the aspect word" (Ma et al. 34).

Although these classification levels are frequently used for MD they carry some drawbacks. The inclusion of "additional information to enhance the model's ability to recognize metaphors" like POS tag sequences or target words to the model design "introduce[] noise, potentially impacting the performance of metaphor recognition" (Song et al. ch. 2). Moreover, Ma et al. found that when training with Transformer-based LLMs these classification levels are "overly simplistic" (34). Therefore, based on the assumption that the metaphoricity of a sentence is not bound to a single aspect word or POS sequence but rather stems from the context of the whole sentence, Ma et al. propose the sentence-level classification (33-34). It is formally defined as "[g]iven a sentence s, predict whether s is metaphorical" (ibid. 34), which means that

the whole sentence serves as the input to the LLM, while any additional feature markings are ignored (ibid 38).

Due to the disadvantages of the POS-level and word-level annotation methods and due to the lack of information whether Kesarwani et. al reannotated TroFi and MOH to match with the POS-level annotation of the PoFo dataset, I chose to adopt the generalised and simplified concept of Ma et al. for all datasets to ensure consistency. Thus, I define the MD task a sentence classification problem. To this end, any feature markings in the datasets are deleted and the sentences are binary labeled as either “metaphorical” or “literal”.

3.2 Transformers Framework: Fine-tuning DistilBERT

The most popular approach to detect metaphors is training on fine-tuned Transformer LLMs, particularly on derivations of the encoder model BERT (Reimann and Scheffler 89). This is why I chose to fine-tune the Transformer-based pre-trained LLM DistilBERT (Sanh et al.), an efficient variant of BERT (Devlin et al.), on the MD task.

For the practical implementation the HuggingFace Transformers framework is used (Wolf et al.). The workflow of the Transformers framework comprises the components tokenizer, transformer and a head (ibid. 39-40). The model-specific tokenizer maps the dataset’s text tokens to indexes, the transformer converts these indexes to contextual embeddings and the pre-trained head uses these contextual embeddings for predictions on general tasks like text classification (ibid.). Finally, the head can be fine-tuned on specific tasks like MD (ibid.).

The advantage of fine-tuning a LLM instead of pre-training it from scratch lies in the concept of transfer learning which saves time and computational resources (“How do Transformers work?”). For transfer learning a language model is initially pretrained on large amounts of data such that it forms “statistical understanding of the language” (ibid.), which serves as a building block for further processes. Then this generalised knowledge can be transferred to a specific downstream task like MD by fine-tuning the model on specialised training data like the metaphor datasets (ibid.). Since the pre-trained model can be reused on numerous tasks, fine-tuning “has lower time, data, financial, and environmental costs” (ibid.), which is why it is used in this paper.

The model for fine-tuning must align with the requirements of the desired task, which in the case of MD is an English text classification task. The encoder LLM DistilBERT was “pretrained on the same data as BERT, which is BookCorpus, a dataset consisting of 11,038 unpublished books and English Wikipedia” (“DistilBERT base model”), which makes it suitable for the fiction and non-fiction sentences from the Graph Project datasets. Moreover,

the method of "knowledge distillation" (Sanh et al. 2) achieves a 40% reduction in model size (from 110M parameters and 12 layers to 67M parameters and 6 layers), 60% more speed compared to BERT while maintaining 97% of BERTs performance (ibid. 1,3). Due to DistilBERT's similarity to BERT which is prominently used for the MD task, its enhanced speed and sustainability it will be used in this paper.

3.3 SetFit Framework: Few-shot fine-tuning with all-MiniLM-L6-v2

While Transformer LLMs are largely used for the MD task, they require training on large datasets which is a challenge for literary metaphor data since "big data is […] a concept that has features and implications that are hardly applicable to literature" (Tanasescu et al. 122). Working with four small datasets this paper wants to address this challenge by leveraging few-shot learning with the framework SetFit, which is designed for small-scale datasets.

Few-shot learning entails training pretrained LLMs with datasets that contain little labelled data (Tunstall et al. ch. 1). To achieve good evaluation results with few-shot learning high parameter count LLMs like GPT-3 or T-FEW, which pose sustainability disadvantages, and manual prompt-engineering, which is time and cost consuming, are typically used (ibid. ch. 2). However, in 2022, Lewis et al. developed SetFit (Sentences Transformer Fine-tuning), which is "an efficient and prompt-free framework for few-shot fine-tuning of Sentence Transformers" (ibid.). In contrast to regular Transformer architectures that are tailored to understand individual text tokens, Sentence Transformers can compare pairs of whole sentences to "derive semantically meaningful sentence embeddings" (Reimers and Gurevych 3982).

To achieve prompt-free few-shot learning SetFit operates in two steps. First, in the "embedding fine-tuning phase" ("Sentence Transformers Finetuning") a Sentence Transformer model, which can be chosen by the user, generates positive or negative pairs of sentences from a dataset.[5] Then the model transforms the sentences into embeddings and fine-tunes these embeddings such that positive embeddings are closer to each other in the vector space (ibid.). This process is called contrastive learning (ibid.). The advantage of using it with small datasets is that it enlarges the dataset significantly by constructing all possible positive and negative pairs whose number "grows exponentially to the number of sentences" (ibid.).

In the "classifier training phase" (ibid.) the sentences from the dataset pass through the newly trained sentence transformer embedding model and are mapped to the right labels by a

[5] In this context positive pairs are two sentences with the same label and negative pairs consist of one sentence with a positive and one sentence with negative label ("Sentence Transformers Finetuning").

logistic regression classifier (ibid.). Since this workflow allows effective and fast results the SetFit framework will be utilized in this paper.

For the few-shot approach with SetFit the all-MiniLM-L6-v2 Sentence Transformer model is chosen based on its evaluation results, which Reimers and Gurevych have published in the Sentence Transformers documentation ("Pretrained Models"). According to their findings, despite the model's compact size of 22.7 M parameters and 6 layers, it is 5 times faster than the best-performing model, all-mpnet-base-v2, while retaining a comparable performance (ibid.). Considering its size, speed, and sustainability all-MiniLM-L6-v2 is more similar to DistilBERT than other Sentence Transformers, allowing for a performance comparison between the Transformers and SetFit frameworks on the Graph Project datasets. The next chapter deals with dataset preprocessing and analysis as well as with the functionality of the Transformers and SetFit framework implementations.

4. Experimental Setup

This chapter describes the setup of the fine-tuning experiments. First, it expounds the data management process and provides an overview of dataset properties such as label balance and sentence quantity. Then the steps to implement the Transformers and SetFit methodologies are explained. Special attention is given to coding parts which deviate from the suggested framework workflow to ensure transparency for potential reproducibility needs.[6]

4.1 Dataset Management: Preprocessing and Data Analysis

For model fine-tuning the data for testing and training needs to be cleaned and transformed into a consistent format. The transformed datasets can then be analysed to gain deeper insights into their properties like sentence quantity and class distribution. Since Kesarwani et al. do not explicitly describe their preprocessing process, it was not possible to confidently replicate their dataset setup. This is why I decided to reuse the cleaned versions of the TroFi and MOH datasets

[6] The project was coded in Python 3.12.1. Coding was done in the Google Colaboratory environment with the intention to employ its TPU v2 access for prototyping the code functionalities and for faster training. However, due to unstable TPU v2 availability the final training and testing was done using an NVIDIA RTX 4090 GPU. The following libraries were used throughout the code: General purpose libraries Pandas (the pandas development team), Numpy (Harris et al.) and Sklearn (Pedregosa et al.); HuggingFace libraries Evaluate, Datasets, Transformers (Wolf et al.) and SetFit (Tunstall et al.); visualisation libraries Seaborn (Waskom) and Matplotlib (Hunter). For runtime and emission information see the ethical concerns chapter.

by Su et al., who publicly published their data for their MD project DeepMet on GitHub.[7] This ensures a standardised starting point and aligns with the idea of research data reusability. For MOH, Su et al. cleaned the sentences by removing redundant quotation marks and adding punctuation where necessary ("MOH-Xnotes"). For TroFi, they deleted unnecessary punctuation marks at the end of the sentences and reshaped the data from a group-wise segmentation of metaphorical and literal samples to a vertical format with the columns "verb, sentence, verb_idx, label". For PoFo I used the original dataset by Kesarwani et al.

The first step in preprocessing was to normalise all datasets into the tab-separated columns "text" and "label" that comprise sample sentences and the labels "metaphorical" and "literal", such that they fit into the definition of MD as a sentence-level classification task (PA 6-9). While the label names for MOH and TroFi were converted from numerical labels to strings (PA 8-9), the label names for PoFo were extracted from the last character of the original label name, where "y" was converted to "metaphorical" and "n" to "literal", excluding the "s" (skipped) label (PA 6:10-17; Kesarwani et al. 3). Next, the aggregated dataset PoFo_TroFi_MOH was created by concatenating the three normalised datasets (PA 10).

The expression `PoFo_TroFi_MOH_df.describe()` shows that among 4946 sentences of the aggregated dataset 4796 are unique which indicates that it contains 150 duplicate sentences (PA 11). Filtering the aggregated dataset further to identify the properties of the duplicates it gets clear that some sentences appear multiple times but with different labels attached (PA 12-13). An explanation for this is that Kesarwani et al. defined MD as a POS-level classification task so that one sentence could include multiple POS sequences which could be classified with different labels. Since I follow the sentence classification approach, I decided to remove all 150 duplicate sentences to avoid training on inconsistent data so that each sentence is classified unambiguously to one label (PA 14). As the last preprocessing step, the normalised datasets are saved in CSV format for reusability (PA 15). Table 1 provides a sample sentence with the label "metaphorical" for each normalised dataset as well as the dataset's domain to highlight content and length differences among datasets.

The dataset analysis provides insights into sentence quantity (PA 16-19), sentence distribution (PA 20) and class distribution (PA 21-25) of the datasets. Table 2 depicts the quantity of sentences for each dataset sorted by dataset source. The discrepancy between the individual sources stem from the preprocessing decisions reported above.

[7] The three necessary dataset files for preprocessing are listed above PA 5. The preprocessing outputs four normalised datasets whose names are stated in PA 15.

dataset	domain	sample sentence
PoFo	Poetry Foundation	love is a fiction i must use ,
TroFi	Wall Street Journal	And most of the jobs will go to candidates sought out by search firms , not those knocking on headhunter doors
MOH	WordNet	Her husband often abuses alcohol .

Table 1: Dataset domains and one sample sentence from PoFo, TroFi and MOH with the label “metaphorical”.

	original dataset	Kesarwani et al.	Su et al.	this paper
PoFo	720	680	-	552
TroFi	3739	3542	3739	3603
MOH	1640	646	646	641
PoFo_TroFi_MOH	4870	4870	-	4796

Table 2: Number of sentences for each dataset as reported in the original publications in which these datasets were created, in the paper by Kesarwani et al. and in this paper. Additionally, the number of sentences from TroFi and MOH used by Su et al. are given.

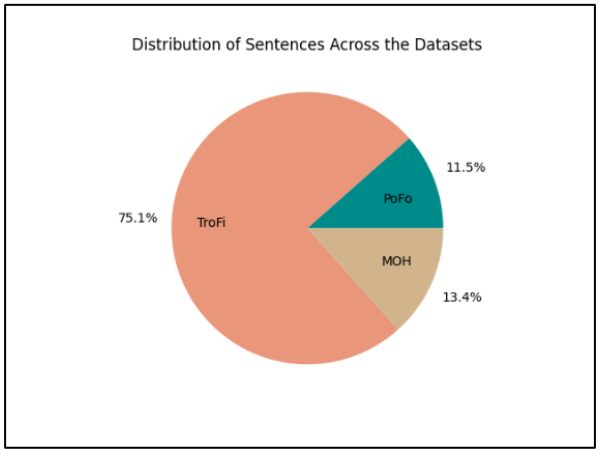

Figure 1: Distribution of sentences from the PoFo, TroFi and MOH datasets inside the PoFo_TroFi_MOH dataset in percent.

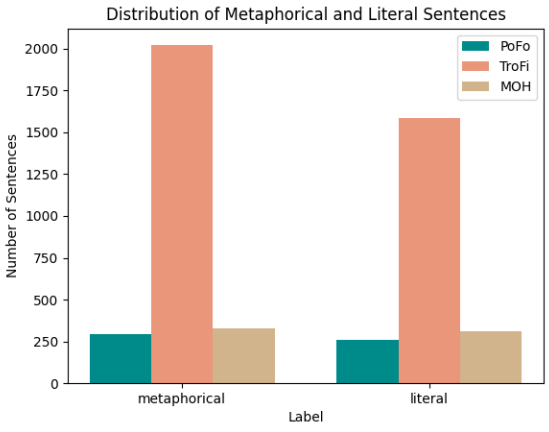

Figure 2: Number of sentences that are labelled “metaphorical” and “literal” for the PoFo, TroFi and MOH datasets.

Figure 1 illustrates the percentage of the sentence distribution of the individual datasets in the aggregated PoFo_TroFi_MOH dataset, showing that most sentences are from TroFi (75.1%). Overall, the datasets contain a small number of samples, raising the question whether this will pose a disadvantage for fine-tuning with DistilBERT and if utilizing SetFit can improve training results. The distribution of classes is a crucial factor for LLM training to avoid biases and overfitting. Figure 2 shows that while PoFo and MOH have fairly distributed labels, TroFi suffers from a label imbalance of about 12% in favour of the literal examples (PA 21-23). The aggregated dataset is slightly unbalanced, with a higher percentage of literal examples (24).

4.2 Transformers Implementation

The fine-tuning of DistilBERT on the normalised datasets was performed with the help of the HuggingFace Transformers framework (Wolf et al.). The code structure and steps of the fine-tuning process were adapted from the Transformers "Text classification" guideline. This chapter provides an overview of this process, while focusing on the deviations from the guideline which were necessary for the MD task and dataset setup. The code is designed to be run separately for each dataset by uncommenting the desired dataset, resulting in four fine-tuned models (TT 4). The Transformers fine-tuning procedure consists of data splitting and preprocessing, evaluation metric definition, setup of training arguments and training. Afterwards the results are saved and documented for future use.

After loading the normalised dataset, it was partitioned into 80% training samples and 20% test samples using the `train_test_split()` function from Datasets (TT 5). This function shuffles and seeds the data by default to guarantee randomness ("train_test_split"). It is important to note that while Kesarwani et al. split the data into two equally large batches (5), I chose a larger training set to provide more learning opportunities (TT 5).

For preprocessing, the DistilBERT tokenizer and the dictionaries for binarised labels are initialised (TT 7-8). In contrast to the guideline, I added a list comprehension to the `preprocess_function()` to use the previously created `label2id` dictionary to convert the labels into the numerical values 0 for "literal" and 1 for "metaphorical" (TT 9:2). The `preprocess_function()` returns the tokenized text and truncates the sentences to DistilBERT's maximum input length of 512 tokens (TT 9:3; "Text classification"). Then the entire dataset is tokenised by passing the `preprocess_function()` to the `map` function (TT 10:1). The final preprocessing step involves the `DataCollatorWithPadding` object, which creates batches and "dynamically pad[s]" ("DataCollatorWithPadding") them during collation to save computing

resources by only padding to the longest sequence in the batch rather than the longest sequence in the entire dataset (TT 11; “Text classification”).

While the Transformers guideline employs only the accuracy metric for evaluation, I choose to evaluate on accuracy, precision, recall and the F1 score since these metrics are used by Kesarwani et al, whose evaluation results I want to compare with the Transformers and SetFit results (TT 12; “Text classification”). To define multiple evaluation metrics in the `compute_metrics()` function the `combine` function from the library `evaluate` is used (TT 12:2; “CombinedEvaluations”).

For fine-tuning, the DistilBERT model is selected (TT 13), the training arguments are specified (TT 15:3-15) and the `Trainer` is initialised (TT 15:17-25) so that `trainer.train()` (TT 15:17) fine-tunes the model. After some initial experiments I set the batch size to 32, the learning rate to 2e-5 since it yielded better results than the default 5e-5 and the training-epochs to 5, which was better than the default 3. The training is replicable with the seed 42.

To work with the results at a later stage I generated a CSV file with the evaluation metrics in a DataFrame for each dataset and saved the fine-tuned models (TT 17-18). A text file documents the evaluation metrics (TT 19).

4.3 SetFit Implementation

This section demonstrates the implementation of the SetFit framework using the Sentence Transformer model all-Mini-LM-L6-v6. The SetFit Quickstart documentation served as a guideline for the basic code structure with additional segments added to fit the MD task (“Quickstart”). These additions will be illustrated with more detail. Opposite to the Transformers implementation, which needs to be rerun for each dataset, the for-loop in the training part of the code allows fine-tuning the models one after the other automatically (ST 7:1). This modification poses a more efficient approach but makes the fine-tuning process less adaptable for alterations due to the extended runtime. For clarity, status print-statements were added to inform the user about the training progress and outputs.

Before training can start, the normalised datasets and the model have to be loaded and the evaluation metrics must be defined (ST 4-6). For both framework evaluation results to be comparable I chose the same evaluation metrics as for the Transformers implementation (ST 6). The technical difference is that the function for defining multiple metrics “must take two arguments (`y_pred, y_test`) and return a dictionary with metric keys to values” (“setfit.Trainer”) which I implemented in the `compute_metrics()` function (ST 6).

The training part of SetFit is much simpler than Transformers since there is no need to manually add a collator, tokenise sentences and binarise labels. Instead, the dataset is split into 20% held-out data and 80% training data (ST 7:9-11), the hyperparameters are set (ST: 7:17-22) and the trainer is initialised (ST 7:25-31) so that the model can be fine-tuned by calling `trainer_train()` (ST 17:33).[8] Originally, I planned to pinpoint the most effective hyperparameters with the hyperparameter optimisation software Optuna (Akiba et al.), which searches a user predefined optimisation space and outputs the best hyperparameter settings. However, after attempting this approach I decided against it since I found that the small increase in performance did not justify the high computational cost it produces by examining all possible parameter combinations. Additionally, research indicates that hyperparameter tuning can lead to model overfitting, potentially worsening performance on unseen data (Rosa et al. 2). Therefore, I manually chose the number of epochs as 5 and the batch size as 32 like in the Transformers setting (ST 7:18-19).

To record the outputs of the SetFit implementation I saved the fine-tuned models (ST 7:40-41), transformed the evaluation metric values from the `evaluation_results` dictionary into a DataFrame for future use (ST 7:46-48; 54-55) and documented the evaluation results for reporting purposes in a text file (ST 7:61-65). The results of the Transformers and SetFit frameworks will be discussed in the next chapter.

5. Experiment Results

Table 3 presents the evaluation results of fine-tuning DistilBERT with the Transformers approach and fine-tuning the Sentence Transformer all-MiniLM-L6-v2 with the SetFit method on the MD task at sentence-level (EV 37). A visualisation of these results is displayed in Figure 3 (EV 40). I chose the F1 score to compare results since it is a metric that gives "a single score that balances both the concerns of precision and recall [and it is] particularly useful when dealing with imbalanced datasets" (Zhao and Fan 306), which applies to the datasets as the data analysis illustrated. The baseline for result comparisons is the rule-based and statistical approach by Kesarwani et al. All reported results are from evaluating the fine-tuned models on the test splits.[9]

[8] It is interesting to consider the impact of the generated sentence pairs by the Sentence Transformer on the dataset size: From 441 samples of the PoFo train dataset SetFit generated 98366 unique pairs (ST 7 output). This demonstrates how the SetFit architecture significantly enlarges a small dataset.

[9] For a comprehensive overview of the results for all metrics see Appendix A.

Overall, the LLM fine-tuning approaches proposed in this paper outperformed the baseline on all datasets except for TroFi. For PoFo there was a significant performance increase of 12.41% with SetFit (F1 0.752) and 2.84% with Transformers (F1 0.688). The MOH dataset displayed an F1 score of 0.785 with the Transformers approach (0.77% increase). Noteworthily, the SetFit approach demonstrated a significant improvement with an F1 score of 0.862 (10.37% increase) on the aggregated dataset PoFo_TroFi_MOH over the baseline. However, the TroFi dataset performed much better with the rule-based and statistical method, surpassing SetFit by 22.25%.

As for the comparison of Transformers with SetFit, SetFit achieved better performance on PoFo, TroFi and the aggregated dataset, while Transformers gave better results on the MOH dataset. Overall, no method excelled across all datasets, but the LLM approaches generally performed better. The next chapter will discuss possible reasons for these results and their implication for the effectiveness of contextual word embeddings in detecting literary metaphors.

approach	**PoFo**	**TroFi**	**MOH**	**PoFo _TroFi _ MOH**
Kesarwani et al.	0.669	**0.827**	0.779	0.781
Transformers	0.688	0.620	**0.785**	0.631
SetFit	**0.752**	0.643	0.734	**0.862**

Table 3: F1 scores for the PoFo, TroFi, MOH and PoFo_TroFi_MOH datasets as reported by Kesarwani et al. compared to the F1 scores achieved in this paper during the fine-tuning of the Transformers and SetFit models. The highest F1 scores for each dataset are marked in bold.

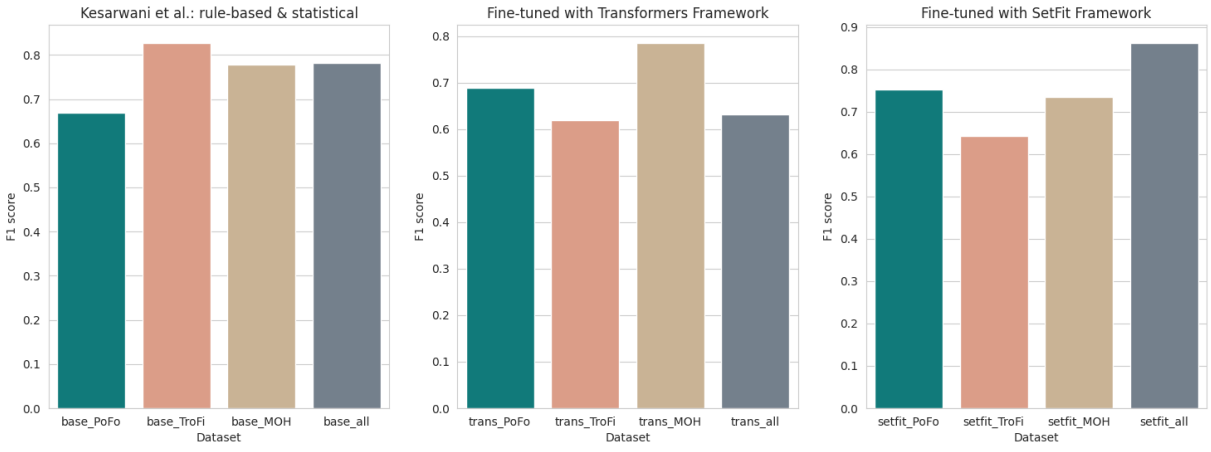

Figure 3: Visualisation of the F1 scores of each dataset subdivided by the approaches rule-based and statistical, fine-tuning with Transformers and fine-tuning with SetFit.

6. Discussion

This paper analyses the question whether fine-tuning with LLMs increases the performance on the MD task compared to rule-based and statistical approaches. Additionally, it examines whether using a few-shot learning approach like SetFit yields better results than ordinary text classification with Transformers considering the small size of the datasets. The results indicate a tendency that fine-tuning achieves better performance than the results reported by Kesarwani et al. Their highest F1 score for the aggregated dataset trained with traditional machine learning approaches was 0.781 and with convoluted neural networks 0.833 (Kesarwani et al. 5, Tanasescu et al. 124), while SetFit produced a F1 score of 0.862 (Table 3). Furthermore, the findings support the assumption that SetFit performs better than the Transformers framework. The performance improvement of 12.4% on the PoFo dataset with SetFit compared to the baseline underlines the usefulness of few-shot training for literary metaphor.

Despite these promising outcomes fine-tuning did not achieve better performance on TroFi and the SetFit methodology fell short on MOH compared to Transformers. Thus, no definitive recommendation can be made for using fine-tuning for MD on these datasets. Before looking at possible explanations for performance differences it should be noted that this paper introduced variations in dataset setup compared to the baseline approach by Kesarwani et al. Specifically, they did not provide relevant dataset preprocessing processes regarding methods to join the datasets and whether MOH and TroFi were reannotated to fit their MD task definition. Consequently, this paper did not replicate their experiment setup but used their results as a comparative baseline. These circumstances should be considered when judging the comparability of results.

Since the ambiguous results occurred between the datasets and not between the fine-tuning methodologies, the first assumption is that this could be due to dataset characteristics. The heterogeneity and diversity of the aggregated dataset regarding the length, number and complexity of sentences and different metaphor domains could have prevented overfitting and established broader contextual information which may explain the resulting highest F1 score. Kesarwani et al. considered these characteristics as advantageous for their results of the aggregated dataset (5,7).

On the other hand, these factors could also pose challenges for fine-tuning the individual datasets. For instance, TroFi's low performance could stem from training difficulty caused by the sentence-level approach. Ma et al. find that the sentence-level approach is more difficult for datasets with complex sentences since the context of the whole sentence must be considered

(38-39). Since TroFi contains complex and the longest sentences the complexity of the sentence-level approach could have impeded the model's performance. The label imbalance could have also played a role (Figure 2). However, these arguments can be refuted by the fact that TroFi constitutes 75.1 % of the aggregated dataset (Figure 1). This indicates that TroFi's characteristics are unlikely to be as detrimental to its poor performance since TroFi's substantial representation in the aggregated dataset must have significantly contributed to the aggregated dataset's exceptionally good results.

Ptiček and Dobša found that although researchers have been using TroFi and MOH for MD, the purpose of these datasets is to "annotate linguistic metaphors" and not to "annotate a metaphor for […] machine identification" (12), which leads to the assumption that TroFi and MOH might be inappropriate datasets for the MD task. However, this idea is rebutted by the fact that Ma et al. achieved an F1 score of 94.45 on TroFi with the sentence level approach using BERT (37), which shows that successful training on the MD task is possible.

These considerations lead to the conclusion that the suboptimal performance on TroFi cannot be fully explained by dataset characteristics alone. The chosen LLM models or chosen hyperparameters might also contribute to this result. Although DistilBERT and all-Mini-LM-L6-v6 are faster than larger models and are said to "retain[] 97% of BERT performance" (Sanh et al. 3) and to have "good quality" (Reimers and Gurevych, "Pretrained Models") respectively, they might not generalise well on a domain-specific task like MD. Moreover, these models, being relatively small with six layers, might not possess enough contextual information which might be necessary for MD since research found that "higher layers recognize more complex features" (Dobson 442). Given computational possibilities one could test this assumption by fine-tuning the datasets with SetFit on state-of-the-art Sentence Transformer models like all-mpnet-base-v2 (Reimers and Gurevych, "Pretrained Models") and note any performance changes. Since SetFit enlarges the dataset significantly dataset size should not be an issue.

However, the core difficulty of finding concrete explanations for the results is the interpretability limitation of LLMs. Due to "complicated architectures and opaque decision-making mechanisms" (Dobson 431) LLMs are called "black-box models" (ibid.). Contrary to statistical machine learning approaches no information is given about which features contribute most to the classification task during fine-tuning (ibid. 445). Thus, it is not clear on which accounts certain samples were classified as "metaphorical" or "literal". The interpretability limitation is especially concerning for the Digital Humanities, where hermeneutics is a central objective for research (ibid. 433). Thus, Dobson urges humanities scholars to "demystify and

critique these technologies" (434), a task to which this paper contributes. Possible reasons for the performance of the models could be discovered by ablation techniques and visualisations of attention weights to observe the role of tokens for model output (ibid. 443-445).

Ultimately, the black-box nature of LLMs makes it challenging to find explanations for TroFi's poor and simultaneously the aggregated dataset's excellent performance. Possible reasons include the complexity of the sentence-level approach, the specific characteristics of the TroFi dataset and used models. Employing larger models for the SetFit approach could potentially lead to an overall performance increase for all datasets.

7. Ethical Concerns

This chapter reports on the replicability, transparency and environmental impact of the research done for this paper as well as the accessibility of used datasets and models. The notebooks with the code for this paper include detailed commentary which supports replicability and reusability for other researchers. However, no specialised code environment was created so it is possible that updates of packages my affect running the code in the future. To ensure transparency the code, outputs and datasets are publicly available on GitHub and the models on Zenodo, which allows building upon the findings of this paper. All used datasets and models are publicly available and have licenses that allow reuse for research purposes.

To track the computational resources, I employed the CodeCarbon software by Courty et al. during the training of the Transformer models (for the functionality see for example TT 14,16). CodeCarbon records the carbon footprint "measured in kilograms of CO_2-equivalent per kilowatt-hour" (Courty et al. "CodeCarbon Methodology"). Appendix B shows that the environmental impact of training DistilBERT was 9.238×10^{-3} CO_2eq for 2.98 minutes and for all-MiniLM-L6-v2 2.145 CO_2eq for 15.18 hours.[10]

[10] The reason for the high emissions for SetFit is likely caused by the significantly enlarged datasets that are created during the embedding fine-tuning phase in the SetFit methodology. Still, this emission result is very low compared to fine-tuning state-of-the-art models like BERT-base, which consume 1438 CO_2e lbs (652 kg) in 79 hours (Strubell et al. 3648). However, such experiments cannot be regarded as direct comparisons since they are conducted with different hardware, machines, dataset sizes, task types, hours trained and hyperparameters.

8. Conclusion

This paper has investigated the MD task from a Digital Humanities perspective by applying state-of-the-art NLP methodology of LLM fine-tuning and few-shot learning on literary metaphors. Building on the research of Kesarwani et al. and Tanasescu et al. this paper compared the outcomes of their rule-based and statistical approach with fine-tuning of the Transformer-based model DistilBERT and the Sentence Transformer-based model all-MiniLM-L6-v2. The datasets used were PoFo (poetry corpus), TroFi (newspaper corpus), MOH (lexical database) and an aggregated dataset. MD was defined as a sentence classification problem with the labels "metaphorical" and "literal".

Two hypotheses were proposed: first, that the fine-tuning approach would outperform the baseline methods and second, that few-shot learning with SetFit would yield better F1 scores than the classical fine-tuning approach. The results suggest a tendency for these two assumptions to hold true due to the significant performance increases for the fine-tuning approach over the baseline and the SetFit methods over the Transformers approach, especially for PoFo and the aggregated dataset. However, since this improvement was not observed for all datasets, it is impossible to formulate generalised statements on the superiority of fine-tuning over traditional machine learning approaches for the MD task.

The TroFi dataset showed poor results which could stem from its characteristics like sentence complexity and label imbalance or from the small model sizes. In this context this paper argued that due to interpretability limitations of LLMs and their black-box nature it is difficult to confidently determine the exact reasons for the inconsistent results. These findings emphasise that while fine-tuning LLMs is a popular choice for classification tasks and linguistic MD in NLP, it does not necessarily guarantee improved performance and might not be the right tool for Digital Humanities scholars, who prioritise an interpretability of results. Ultimately, it is essential to experiment with different methodologies to identify the most effective approach.

Further work needs to be performed to establish whether larger models could optimise the work done in this paper. The fine-tuned models could be deployed to build interactive tools to provide an innovative perspective to teaching and studying metaphors in educational settings. Future studies on MD could focus on creating larger datasets or examine the effectiveness of prompt engineering. Another promising research direction would be to transition from metaphor detection to metaphor recognition by creating a methodology that finds which tokens of a sentence are metaphorical. All in all, metaphor detection remains an intriguing task with numerous research possibilities in the Digital Humanities.

9. Works Cited

Akiba, Takuya, et al. “Optuna: A Next-generation Hyperparameter Optimization Framework.” *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, edited by Ankur Teredesai, pp. 2623–31, https://doi.org/10.1145/3292500.3330701.

“all-MiniLM-L6-v2.” *HuggingFace,* huggingface.co/sentence-transformers/all-MiniLM-L6-v2.

Birke, Julia, and Anoop Sarkar. "A clustering approach for nearly unsupervised recognition of nonliteral language." *11th Conference of the European chapter of the association for computational linguistics*, 2006, p. 329-336, aclanthology.org/E06-1042, natlang.cs.sfu.ca/software/trofi.html.

Courty, Benoit, et al. “CodeCarbon Methodology.” *CodeCarbon,* mlco2.github.io/codecarbon/methodology.html.

---. “Mlco2/codecarbon: V2.4.1.” Zenodo, 2024, https://doi.org/10.5281/ZENODO.11171501.

“CombinedEvaluations.”*HuggingFace*, huggingface.co/docs/evaluate/package_reference/main_classes#evaluate.combine.

Dankin, Lena, et al. “Can Yes-No Question-Answering Models Be Useful for Few-Shot Metaphor Detection?” *Proceedings of the 3rd Workshop on Figurative Language Processing (FLP),* edited by Debanjan Ghosh et al. Association for Computational Linguistics, 2022, pp. 125–30. https://doi.org/10.18653/v1/2022.flp-1.17.

“DataCollatorWithPadding.” *HuggingFace*, huggingface.co/docs/transformers/v4.40.2/en/main_classes/data_collator#transformers DataCollatorWithPadding.

“Datasets.” *HuggingFace*, huggingface.co/docs/datasets.

Devlin, Jacob, et al. “BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding.” *Proceedings of the NAACL-HLT 2019,* pp. 4171–86, arxiv.org/pdf/1810.04805.

“DistilBERT Base Model.” *HuggingFace*, huggingface.co/distilbert/distilbert-base-uncased.

Dobson, James E. “On Reading and Interpreting Black Box Deep Neural Networks.” *International Journal of Digital Humanities,* vol. 5, no. 2, 2023, pp. 431–49, https://doi.org/10.1007/s42803-023-00075-w.

“Google Colaboratory.” *Google,* colab.research.google.com/.

Harris, Charles R., et al. “Array Programming with NumPy.” *Nature,* vol. 585, no. 7825, 2020, pp. 357–62, https://doi.org/10.1038/s41586-020-2649-2.

"How Do Transformers Work?" *HuggingFace*, huggingface.co/learn/nlp-course/en/chapter1/4.

Hunter, John D. "Matplotlib: A 2D Graphics Environment." *Computing in Science & Engineering,* vol. 9, no. 3, 2007, pp. 90–95, https://doi.org/10.1109/MCSE.2007.55.

Jia, Kaidi, and Rongsheng Li. "Enhancing Metaphor Detection Through Soft Labels and Target Word Prediction." https://doi.org/10.48550/arXiv.2403.18253.

Kesarwani, Vaibhav, et al. "Metaphor Detection in a Poetry Corpus." *Proceedings of the Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature,* edited by Beatrice Alex et al. Association for Computational Linguistics, 2017, pp. 1–9, https://doi.org/10.18653/v1/W17-2201.

---. "PoFo Corpus." www.site.uottawa.ca/~diana/resources/metaphor/type1_metaphor_annotated.txt.

Lakoff, George, and Mark Johnson. *Metaphors We Live By*. University of Chicago Press, 2003.

Ma, Weicheng, et al. "Improvements and Extensions on Metaphor Detection." *Proceedings of the 1st Workshop on Understanding Implicit and Underspecified Language,* edited by Michael Roth et al- Association for Computational Linguistics, pp. 33–42, https://doi.org/10.18653/v1/2021.unimplicit-1.5.

Mohammad, Saif, et al. "Metaphor as a Medium for Emotion: An Empirical Study." *Proceedings of the Fifth Joint Conference on Lexical and Computational Semantics,* 2016, pp. 23–33. https://doi.org/10.18653/v1/S16-2003.

Neuman, Yair, et al. "Metaphor Identification in Large Texts Corpora." *PloS one,* vol. 8, no. 4, 2013, 1-9, https://doi.org/10.1371/journal.pone.0062343.

The Pandas Development Team. *pandas-Dev/pandas: Pandas*. v2.2.2. Zenodo, 2024, https://doi.org/10.5281/zenodo.3509134.

Pedregosa, Fabian, et al. "Scikit-Learn: Machine Learning in Python." *Journal of Machine Learning Research,* vol. 12, no. 85, 2011, pp. 2825–30, https://doi.org/10.48550/arXiv.1201.0490.

Ptiček, Martina, and Jasminka Dobša. "Methods of Annotating and Identifying Metaphors in the Field of Natural Language Processing." *Future Internet,* vol. 15, no. 6, 2023, 1-28, https://doi.org/10.3390/fi15060201.

"Quickstart." *HuggingFace*, huggingface.co/docs/setfit/quickstart.

Reimers, Nils, and Iryna Gurevych. "Pretrained Models." *Sentence-Transformers Documentation*, sbert.net/docs/pretrained_models.html.

---. “Sentence-BERT: Sentence Embeddings Using Siamese BERT-Networks.” *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP),* 2019, pp. 3980–90, https://doi.org/10.18653/v1/D19-1410.

Reimann, Sebastian, and Tatjana Scheffler. “When Is a Metaphor Actually Novel? Annotating Metaphor Novelty in the Context of Automatic Metaphor Detection.” *Proceedings of The 18th Linguistic Annotation Workshop (LAW-XVIII*), edited by Sophie Henning*,* 2024, pp. 87–97, aclanthology.org/2024.law-1.9/.

Reinig, Ines, and Ines Rehbein. “Metaphor Detection for German Poetry.” *Preliminary Proceedings of the 15th Conference on Natural Language Processing (KONVENS 2019),* pp. 149–60, nbn-resolving.de/urn:nbn:de:bsz:mh39-93163.

Rosa, Guilherme Moraes, et al. “To Tune or Not to Tune?” *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Law*, edited by Juliano Maranhão. Association for Computing Machinery, 2021, pp. 295–300, https://doi.org/10.48550/arXiv.2202.03120.

Sanh, Victor, et al. “DistilBERT, a Distilled Version of BERT: Smaller, Faster, Cheaper and Lighter.” *ArXiv,* abs, 2019, pp. 1–5, https://doi.org/10.48550/arXiv.1910.01108.

“Sentence Transformers Finetuning.” *HuggingFace*, huggingface.co/docs/setfit/conceptual_guides/setfit.

“setfit.Trainer.” *HuggingFace*, huggingface.co/docs/setfit/v1.0.3/en/reference/trainer#setfit.Trainer.

Song, Ziqi, et al. “Multi-Task Metaphor Detection Based on Linguistic Theory.” *Multimedia Tools and Applications,* 2024, https://doi.org/10.1007/s11042-023-18063-1.

Strubell, Emma, et al. “Energy and Policy Considerations for Deep Learning in NLP.” *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics,* edited by Anna Korhonen et al. Association for Computational Linguistics, 2019, pp. 3645–50, https://doi.org/10.18653/v1/P19-1355.

Su, Chuandong, et al. “DeepMet: A Reading Comprehension Paradigm for Token-Level Metaphor Detection.” *Proceedings of the Second Workshop on Figurative Language Processing,* edited by Beata Beigman Klebanov et al. Association for Computational Linguistics, 2020, pp. 30–39, https://doi.org/10.18653/v1/2020.figlang-1.4, github.com/YU-NLPLab/DeepMet/tree/master.

---. “MOH-Xnotes.” *GitHub,* github.com/YU-NLPLab/DeepMet/tree/master/data/MOH-X/MOH-Xnotes.txt.

Tanasescu, Chris, et al. “Metaphor Detection by Deep Learning and the Place of Poetic Metaphor in Digital Humanities.” *The thirty-first international flairs conference,* 2018, pp. 122–27, aaai.org/papers/122-flairs-2018-17704/.

"Text Classification." *HuggingFace,* huggingface.co/docs/transformers/tasks/sequence_classification.

Toker, Michael, et al. *A Dataset for Metaphor Detection in Early Medieval Hebrew Poetry.* arXiv preprint, 27 Feb. 2024, tokeron.github.io/metaphor/.

"train_test_split." *HuggingFace,* huggingface.co/docs/datasets/v2.18.0/en/package_reference/main_classes#datasets.Dataset.train_test_split.

Tunstall, Lewis, et al. "Efficient Few-Shot Learning Without Prompts." *36th Conference on Neural Information Processing Systems (NeurIPS 2022),* 2022, pp. 1–14, https://doi.org/10.48550/ARXIV.2209.11055.

Vaswani, Ashish, et al. "Attention Is All You Need." *Advances in Neural Information Processing Systems,* vol. 30, 2017, pp. 1–11, https://doi.org/10.48550/arXiv.1706.03762.

Waskom, Michael. "Seaborn: Statistical Data Visualization." *Journal of Open Source Software,* vol. 6, no. 60, 2021, p. 3021, https://doi.org/10.21105/joss.03021.

Wolf, Thomas, et al. "Transformers: State-of-the-Art Natural Language Processing." *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations,* edited by Qun Liu and David Schlangen. Association for Computational Linguistics, pp. 38–45, https://doi.org/10.18653/v1/2020.emnlp-demos.6.

Zhao, Fengxiang, and Fan Yu. "Enhancing multi-class news classification through BERT-augmented prompt engineering in large language models: a novel approach." *Problems and Prospects of Modern Science and Education,* 2024, pp. 297–309, https://doi.org/10.46299/ISG.2024.1.10.

Appendix

Appendix A: Overview of fine-tuning results for Transformers and SetFit

dataset	framework	F1	precision	recall	accuracy
PoFo	Transformers	0.688	0.597	0.811	0.649
	SetFit	**0.752**	**0.653**	**0.887**	**0.721**
TroFi	Transformers	0.620	**0.687**	0.565	**0.691**
	SetFit	**0.643**	0.615	**0.674**	0.666
MOH	Transformers	**0.785**	**0.803**	**0.768**	**0.775**
	SetFit	0.734	0.797	0.681	0.736
PoFo _ TroFi _ MOH	Transformers	0.631	0.691	0.581	0.688
	SetFit	**0.862**	**0.889**	**0.837**	**0.877**

Table 4: Results for the evaluation metrics F1 score, precision, recall and accuracy sorted by dataset and fine-tuning method. The best performing method for each metric is highlighted in bold.

Appendix B: Overview of fine-tuning emissions

	DistilBERT	all-MiniLM-l6-v2
Emissions [CO_2eq] in kg	0.0013816308201039678	0.0209728763120426
	0.0035116309184925524	0.7704291161175701
	0.0014721186923761533	0.0134567044835949
	0.002873104610685966	1.3402467884240712
Sum	**0.009238485042**	**2.145105485**
Duration in seconds	6	611.6328690052032
	88.77965497970581	19915.516769886017
	9.583139181137085	535.1892156600952
	74.23649764060974	33595.98231673241
Sum	**178.5992918**	**54658.32117**

Table 5: Emissions gathered with CodeCarbon in CO_2eq in kg for each dataset for fine-tuning DistilBERT and all-MiniLM-l6-v2. The sum of emissions for each model is highlighted in bold. Additionally, the duration in seconds of fine-tuning both models is given for each dataset and as a sum for each model. The data was collected from the individual emission reports for each dataset which can be found in the folders transformers_training_output and setfit_training_output.