BERT model for Metaphor Detection using the Kontextbruch approach

Sergio Solmonte
sergio.solmonte@uni-bielefeld.de

Marco Madonna
marco.madonna@uni-bielefeld.de

Abstract

Metaphors detection is still a complex task for machines and achieving considerable results would be a step forward in the evolution of artificial intelligence. In this work several models based on the same BERT (Bidirectional Encoder Representations from Transformers) architecture have been trained and evaluated, obtaining encouraging results in the task of metaphors detection (F1-score of 0.31 and Recall of 0.47 for metaphors). To do this, a German dataset was used, manually annotated following the *Context Breach* approach. Even if we get better results than previous experiments, our attention was focused on the different possible configurations of parameters, in order to provide guidelines for future implementations that want to use an architecture similar to the proposed one based on BERT.

1 Introduction

The correct understanding of metaphors represents a complete mastery of human communication and language, detecting metaphors is still a very challenging task in the NLP community.

Correctly identifying metaphors across different languages, is not an easy task even for human annotators. In order to define what a metaphor is, and how to detect it, multiple theories were formulated in the linguistics community and several NLP approaches can be derived from them, as shown in Section 2.

In this work, we start from a dataset, composed of several German text fragments manually annotated following the *Context Breach* approach, described in Section 3. We considered applying a strong pre-processing on the data, but this approach will not allow us to understand the real differences between German metaphors and non-metaphors. Hence, we only choose to perform a *Data Augmentation* in order to increase the number of relevant texts. More details are in Section 4.1.

In our approach, described in Section 4 we expand on previous work, done by students, by *fine-tuning* a pre-trained BERT model, trying to obtain better results compared to previous works. Previous students focused their attention on how a different pre-processing on the input data can improve the performance of different BERT models and on how to model this metaphor recognition task. Considering the desired output, which involves recognizing three distinct classes (metaphors, non-metaphors, and metaphor candidates) using a labeled dataset, we choose to approach this as a single-task classification problem. We use two different BERT models trained on German texts GBert (Branden Chan, 2022) and HistBert (Huang et al., 2015), to then perform classification on their output, analyzing different optimization algorithms and loss functions that can be used to improve their performance.

Due to the class imbalance intrinsic to the problem, in any text metaphors will be far less than non-metaphors, we conduct our experiments (described in section 5) by calculating the F1-scores and the accuracy scores. We also checked the code done by previous students to recreate their results and compare our own results to theirs.

We then propose and evaluate different *hyperparameters* configurations for the final network, that is used to classify the output of the BERT model, as well as the results with and without *data augmentation*. See *Ablation Study* (6) for more details about the hyperparameters choice.

Our best results are referred to the last model, called *GBert8*, that obtains an F1-score on metaphors of **0.31**, an F1-score on non-metaphors of **0.96** and an F1-score on metaphors candidates of **0.44**, with an F1-score macro average of **0.57**. All the results are an average over the different folds used during the evaluation explained in Section 4.5.

2 Related works

Detecting metaphors is a daunting task for natural language processing, most languages use them in different ways, and the ever-changing meaning and context of words make it a difficult task even for humans.

Over the years, various metaphor theories have been proposed to provide a more comprehensive definition. Below are some of the most influential ones:

The **comparison** theory suggests that metaphors involve the comparison of two things that share some similarities, but are also different in some ways, so metaphors are not just decorative, they help us to understand abstract concepts by relating them to concrete experiences.

The **interaction** theory suggests that metaphors involve the interaction of two cognitive domains, such as the concrete and the abstract, so metaphors involve mental processes.

The **cognitive** theory suggests that metaphors involve the use of mental models to understand and make sense of the world and according to this theory metaphors also involve mental representations of the world.

The **embodiment** theory suggests that metaphor involves the use of bodily experiences and sensations to understand abstract concepts and according to this theory, they involve embodied experiences.

The **conceptual** metaphor theory proposed by George Lakoff (Lakoff and Johnson, 1980), suggests that metaphor is not just a linguistic device, but it is a fundamental mechanism of human thought and according to this theory, metaphors are not just used to express abstract ideas, but they are also used to structure our conceptual system.

Linguistic and in particular the conceptual metaphor theory has different methods to define them and their use. Based on these definitions different ways to identify them can be found. In (Schneider et al., 2022) they consider the metaphorical use of adjectives, focusing on finding the couples adjective/noun, where the two words come from two different semantic domains (like *sweet thought*, *raw emotion*, or *clear answer*), using an unsupervised learning method on a simple feed-forward neural network to find them.

More generally, following the context breach approach, we can identify when a word is used metaphorically by finding the word, or words, that breaks the context, the semantic domain, of the sentence in which it is present. Following this idea, semantic embeddings of words are useful to perform classification tasks on them, in (Do Dinh and Gurevych, 2016) they use word embeddings created with *Word2Vec* to then perform a token by token tagging via a *MultiLayer Perceptron* (MLP). Recently approaches based on BERT have been presented, starting from the fine-tuning of various BERT models to then perform classification tasks based on different theories to detect metaphors like in (Liu et al., 2020), (Choi et al., 2021) and (Li et al., 2023). Building on this and previous works accomplished by students from past years, we applied the context breach approach to German-based BERT models to detect metaphors in German texts.

2.1 BERT Model

BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2018a) is a pre-trained transformer-based model used in Natural Language Processing (NLP), designed to help computers understand the meaning of ambiguous language in a text by using surrounding text to establish context. BERT uses *WordPiece* embeddings (Wu et al., 2016), in which the vocabulary is initialized with individual characters of the language, then the most frequent combinations of symbols in the vocabulary are iteratively added to the internal vocabulary, building a 30,000 token vocabulary.

This *WordPiece* is used to *tokenize* the input, converting the text into a sequence of tokens. The first token of every sequence is always a special classification token (*[CLS]*) and the final hidden state corresponding to this token is used as the aggregate sequence representation for classification tasks. How we use this output is explained in Section 4.2.

After pre-training, BERT can be fine-tuned on a specific task such as Metaphors detection. Fine-tuning involves training the model on a smaller dataset specific to the task, which adjusts the weights of the model to perform well on the new task.

2.1.1 German BERT

The model called *bert-base-german-cased* (Branden Chan, 2022) is pre-trained with a dataset composed of GermanWiki, OpenLegalData and news articles written in German. Evaluating the model using Conll03(Ger), GermEval14 for Named-Entity Recognition and GermEval18, GNAD for sentiment classification tasks. This model per-

formed better on these tasks than the *Multilingual* model (Devlin et al., 2018b) as shown in Table 1 and we have chosen to use it as it currently represents the *State Of the Art* for German texts.

Model	germEval18Fine	germEval18Coarse	germEval14	CONLL03	10kGNAD
Multilingual-cased	0.441	0.71	0.834	0.792	0.888
Multilingual-uncased	0.461	0.731	0.823	0.786	0.901
German BERT - cased	**0.488**	**0.747**	**0.84**	**0.804**	**0.905**

Table 1: German BERT cased results. The model is directly compared with the multilingual one, both the cased and uncased versions. Table from from OpenAI.

2.1.2 Historical BERT

This BERT model was developed in (Huang et al., 2015) with the aim of automatically tagging four representation forms of speech/thought/writing in fictional and non-fictional German historical texts. We consider using this model following previous works done by other students for this metaphor classification task. This model was trained on fictional and non-fictional German texts written between 1840 and 1920 obtaining very good results as shown in Table 2.

Tag	F1	Precision	Recall
Direct	0.80	0.86	0.74
Indirect	0.76	0.79	0.73
Reported	0.58	0.69	0.51
Free indirect	0.57	0.80	0.44

Table 2: Historical Bert Results. The model aims to classify the text into the four possible representations written in the tag column. So we can see it as a classification into 4 classes.

3 Data

The basis of the dataset used in this experiment is the result of the work done by past years' students and it is composed of a set of text fragments extracted from a large corpus of old German texts and hand-labeled by different annotators. Each fragment has a label that designs it as a metaphor, a metaphor candidate or a non-metaphor. Every annotator first labeled the fragments and then the results were jointly reviewed to reach an agreement and create a gold standard, as can be seen in Table 3 and 4.

The resulting dataset contains two sets of the same fragments, one annotated with the gold standard labels and one with repeating fragments associated with the different labels given by different annotators. Since the dataset is relatively small we decided to use only the gold standard part to have better data quality and consistency.

Index	Metaphor	Candidate	Non-metaphor	Gold standard
1	0	0	1	Non-metaphors
2	1	3	0	Candidate
3	3	1	0	Metaphor

Table 4: Example of how the gold standard is used. They first consider all the scores assigned to each class (Metaphor, Candidates and Non-metaphor) by the annotators and next they assign the class that has the maximum score to the text fragment, obtaining a single element composed by *Text* and *Label*.

The gold standard dataset is made of 4658 text fragments and its composition can be seen in Figure 1

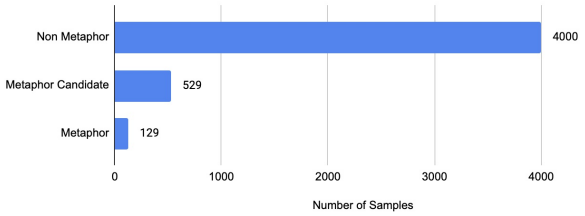

Figure 1: Gold standard dataset composition

To achieve metaphors detection using the *Context Breach* approach we have to find a context breach, which means an unusual choice of words considering the context in which it is used. The context of a sentence is described as the frame while the metaphor that *breaches* the frame is described as the **focus**.

Der politische Körper verwendet in beiden Fällen seine meiste Kraft auf die Zunahme von **Zahnen und Krallen**.

Table 5: Simple example of a metaphor in the dataset. The goal is to recognize if there is something inside this text fragment that is out of context. The frame is underlined and the focus is in **bold**.

Considering the example in Table 5 the frame is politische Körper, which means political body, while the focus is **Zahnen und Krallen**, which means teeth and claws. Our translated metaphor, used solely for better comprehension, is *In both cases, the political body expends most of its energy on the growth of teeth and claws.* The *"growth of teeth and claws"* is out of place in the political context denoted by *"political body"*, and this is what we want our model to learn.

Index	Text	Metaphor	Candidate	Non-metaphor
1	Darauf deutet wenigstens die ursprüngliche Geschlechtsanlage, die bei allen Vertebraten herniophrodiiisch ist.	0	0	1
2	Aber das Licht der Selektionstheorie reicht noch weiter.	1	3	0
3	Der politische Körper verwendet in beiden Fällen seine meiste Kraft auf die Zunahme von Zähnen und Krallen.	3	1	0

Table 3: Examples of manually annotated text fragments before the gold standard application. In the metaphor and candidate columns, we have the number of annotators who deemed that text fragment to be a metaphor or candidate. As far as non-metaphors are concerned, only one annotator needs to negatively evaluate that text fragment to go directly to the next one. The index is reported to better understand Table 4 but it is not provided in the dataset.

Bei den Veteranen von 1848, bei den Alten unter den Sozialdemokraten die nicht von Marx loskommen können, bezeichnet man die häufige Wiederkehr der selben festsitzenden Gedanken als Doktrinarismus; **die älteren Nationalliberalen bleiben am Ufer, während die nationalliberale Jugend auf sozialpolitisches Fahrwasser hinausschwimmt**.

Table 6: More complex example of a metaphor in the dataset. Here we have abstract concepts as frame and **focus**, so this could be more difficult to recognize for a machine. In this case the political ideology is the frame and the **divide between older and younger** is the focus.

In the dataset, we also have more complex metaphors as shown in Table 6. In this example, we have the *political ideology* as the *frame* and the ***divide between older and younger generations*** within political groups and their willingness or reluctance to embrace new ideas as the focus. In this case, the metaphor uses the imagery of swimming and being stuck on the shore to illustrate this divide. This concept differs from more famous metaphor theories where a metaphor is seen as a mapping of a source domain to a destination domain, so it will be more complex to understand and recognize it.

3.1 Preprocessing

Due to the small size of the dataset and the really small presence of metaphors in it, we decided to artificially augment the data by using *Back-translation* described in Section 4.1. Before performing data augmentation, we divide it into a train/validate set and test set so we can test the model on real samples of metaphors and keep it clean from duplicates of the train set fragments that might be introduced through the back translation process. Performing augmentation before splitting the dataset can introduce duplicates that, although different in body, have the same substantial meaning, automatically removing this kind of duplicate gets really complicated. The composition of the test set is shown in Figure 2.

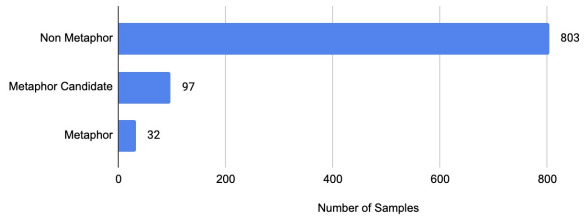

Figure 2: Test set composition. Since no data augmentation is done here, we have a small number of metaphors compared to the number of non-metaphors.

4 Methods

4.1 Data Augmentation

As it is possible to see in Figure 1, the dataset is heavily imbalanced, so the first task is to augment the data with *artificial data*. This data is produced by *Back-Translation* using seven different languages: English, Italian, Polish, Spanish, Danish, Finnish and French. In the perspective of testing the model over unseen and *pure* data, we first want to split the data into two different subsets, called *sub-train* and *sub-test* with a ratio of 80:20 respectively, and then perform data augmentation only on the sub-train because we want to test the model on the original data, without the artificial ones.

The augmentation was performed only on the

Text	Translation
Zentralund Südamerika sind aufgehende Sonnen; Italien eine niedergehende.	Original
Mittel- und Südamerika sind aufgehende Sonnen, Italien eine untergehende.	English-German
Mittel- und Südamerika sind die aufsteigenden Sonnen; Italien ist gesunken.	Italian-German
Mittel-und Südamerika sind aufgehende Sonnen; Italien ist eine, die verschwindet.	Polish-German
Mittel- und Südamerika ist die aufgehende Sonne; Italien ist eine untergehende Sonne.	Spanish-German
Zentral- und Südamerika sind Sonnen, die sich erheben; Italien eine geht hinunter.	**Danish**-German
Zentral- und Südamerika sind Sonnen, die sich erheben; Italien eine geht hinunter.	**Finnish**-German
Mittel- und Südamerika sind die aufgehenden Sonnen und Italien die aufgehenden.	French-German

Table 7: Fragment of a metaphor back-translation. Here is reported only the second step of back-translation, which consists of re-translate the sentence into its original language. For this text, **Danish**-German and **Finnish**-German produced the same result. In the end, the second one is then removed from the augmented dataset because we don't want duplicates.

data labeled as *Kandidat* (Candidates) and *Metapher* (Metaphors), after converting the labels from strings to integers, respectively 0 for non-metaphors, 1 for metaphors and 2 for candidates. Before training, our sub-train set is increased to 2996 candidates and 678 metaphors, as shown in Figure 3.

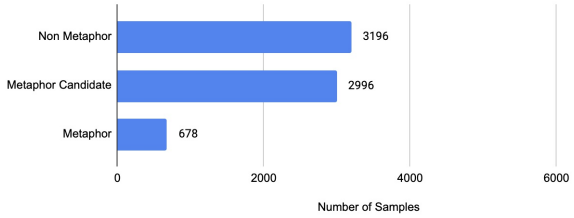

Figure 3: Augmented training set composition. While this set is a bit imbalanced, we now have the ability to use more metaphors in the training process.

4.1.1 Back-Translation

Back translation is a technique used in *Natural Language Processing* (NLP) to generate synthetic data for training machine learning models.

The basic idea behind the back translation is to take a sentence in one language, translate it into another language using an existing translation model, and then translate it back into the original language using a different translation model.

For example, assuming we have a sentence in German that we want to translate into English. We could use a pre-existing translation model to translate the German sentence into English. We would then take the resulting English sentence and translate it back into German using a different translation model. The resulting German sentence should be similar to the original German sentence, but with some variations. Example of back-translation in Table 7.

This technique is particularly useful when training machine learning models for tasks where the availability of large amounts of labeled data is often limited. By generating synthetic data through back translation, it is possible to increase the amount of training data available and improve the performance of a model. Additionally, the resulting synthetic data can introduce a degree of variation and diversity into the training set, which can help the model to generalize better to new, unseen, data.

4.2 Model Architecture

In this experiment, two BERT pre-trained models were fine-tuned and evaluated with several configurations described in Section 5. Both were used as the main component of a Neural Network, following the architecture shown in Figure 4.

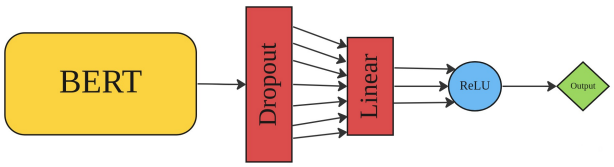

Figure 4: General model architecture. Composed of 2 parts or cores: the BERT head/core and the classifier. The output of the linear layer is a 1×3 vector that represents the three possible classes before the final output class production using an *argmax* function.

For each model, we used its own *tokenizer* to produce the input vectors. After the application of the BERT model, we have two outputs, the first one contains the embedding vectors of all of the tokens in a sequence and it is discarded; the second one, called *pooled_output*, contains the embedding vector of the [CLS] token. It is a vector that first passes into a Dropout layer and next into a Linear layer with *ReLU* as activation function, where we want to classify an input sentence as non-metaphor, candidate or metaphor. Details about optimization and loss function are in Sections 4.3

and 4.4.

4.3 Optimizers

Due to the relatively low dimension of the dataset, we tried two different methods to optimize the training process and obtain the best from it: *Adam* and *AdamW*.

4.3.1 Adam

Adam (Kingma and Ba, 2014), stands for Adaptive Moment Estimation, is an optimization algorithm that can be used instead of the classical stochastic gradient descent procedure, of which it is an extension, to update network weights iteratively based on training data. Stochastic gradient descent maintains a single learning rate for all weight updates and the learning rate does not change during training, with Adam a learning rate is maintained for each network weight (parameter) and separately adapted as learning unfolds. Using per-parameter and adaptive learning rates improves performance on problems with sparse gradients, such as natural language processing problems, and helps to deal whit noisy data.

4.3.2 AdamW

AdamW (Loshchilov and Hutter, 2017), stands for Adaptive Moment Estimation with Weight Decay, is an extension of Adam that, besides using per-parameter adaptive learning rates, adds weight decay directly into the optimization algorithm. Doing this prevents over-fitting of the model by adding a penalty to the loss function based on the model weights.

4.4 Loss function

4.4.1 Cross Entropy Loss

In the final classification task, we have a set of input features X and the target variables Y that for each input can take one of three possible classes: class 0, class 1, or class 2 (following the labels). Our goal is to build a model that can predict the correct class for new input examples. The cross-entropy loss function for this problem is defined as follows:

$$L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=0}^{2} y_{i,k}\, log(p_{i,k}) \tag{1}$$

where N is the number of training examples, k is the index of the class, ranging from 0 to 2 (in our case there are 3 classes) $y_{i,k}$ is a binary indicator variable that is equal to 1 if the true class of example i is k, and 0 otherwise, $p_{i,k}$ is the predicted probability that example i belongs to class k.

In order to obtain the predicted probabilities, we use a *softmax* function in the output layer of the model. The function converts the raw model outputs into a set of probabilities that add to 1.

For a 3-class problem, the *softmax* function is defined as:

$$p_{i,k} = \frac{e^{z_{i,k}}}{\sum_{j=0}^{2} e^{z_{i,j}}} \tag{2}$$

where $z_{i,k}$ is the raw model output for example i and class k. The output of the softmax function gives a set of predicted probabilities for each class, which can then be used to compute the cross-entropy loss function.

4.4.2 Focal Loss

The *Focal Loss* function (Lin et al., 2017) is used to contrast class imbalance, introducing a modulating term to the cross entropy loss function to emphasize learning on difficult misclassified examples. Essentially, the Focal Loss is a cross-entropy loss that scales dynamically, with the scaling factor diminishing as confidence in the correct classification grows. As a result, the contribution of easier examples during training is automatically reduced, allowing the model to concentrate on the more challenging examples.

In math terms, the Focal Loss is created by adding a factor $(1 - p_t)^\gamma$ to the regular cross entropy criterion:

$$FL(p_t) = -(1 - p_t)^\gamma\, log(p_t) \tag{3}$$

By setting this factor to a lower value, the loss for well-classified examples is decreased, while placing more emphasis on the misclassified examples. The parameter γ is called *focusing parameter* and the results of using different γ are shown in Figure 5.

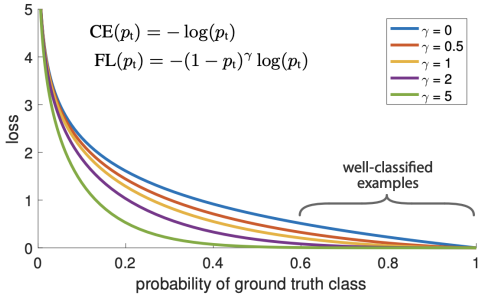

Figure 5: Focal Loss function with different γ. Figure from (Lin et al., 2017).

4.5 Cross Validation

To evaluate and test our models, we applied a *10-fold cross-validation*, making it so that we don't

have test folds without metaphors. This was possible by first separating our test set by class and then dividing each subset into 10-folds, so as not to have empty or inconsistent folds for model evaluation. In order to always have the same folds to test the fine-tuned models, we have set a fixed random state for all the different evaluations. The last operation is to iterate over the number of folds (k=10) and concatenate together the $fold_i$ of the subset that contains elements of class 0 with the $fold_i$ of class 1 and class 2. Before the evaluation, we also shuffle the data randomly so that we don't have continuous sections of the same class. In Table 8 it is possible to see that every fold has approximately the same variance.

fold	1	2	3	4	5	6	7	8	9	10
σ	0.399	0.399	0.397	0.401	0.401	0.401	0.401	0.372	0.372	0.372

Table 8: The variance within each fold between the classes. We can see that each fold has approximately the same variance between elements and this is because we produced each fold ad-hoc, dividing the number of metaphors, candidates and non-metaphors by the number of folds, so that we have the same number of elements in every fold.

5 Experiments

As said before, our attention is focused on finding the best configuration for the architecture proposed in Section 4.2 to obtain satisfying results on the German metaphors detection task.

Our first experiments were done considering the *Historical Bert* (Huang et al., 2015) as the BERT core of our architecture. This model was already trained on a different classification task, so we, as the other groups before us, thought that this BERT model will fit great with the metaphors detection.

The *HistBert*, being trained directly in Old German, needs no pre-training of the BERT core, so only the classifier was trained.

In the second part of our experiments, we considered the newest *GBert* (Branden Chan, 2022) as BERT core. Using this model we are able to compare directly two different evolution of BERT and demonstrate that the metaphors detection task is getting closer and closer to reaching a solution.

The problem with GBert is that, compared to the Historical Bert, it is not trained with Old German texts and poems, so a pre-training of the model is necessary, in order to familiarize it with old German terms that could be considered unfamiliar.

5.1 Different models

In our experimental process, we wanted to obtain a model that could outperform the other ones. We tried several configurations of parameters as loss functions or optimizers since we knew that they may be the best tools for this kind of task.

For this experiment, we developed 4 models with the *Historical* core and 8 models with the *GBert* core. Each model has differences in terms of training epochs, learning rate, optimization function and loss function. All decisions made regarding the parameters are discussed in Section 6. The configurations of the models are shown in Table 9 and it is possible to see that the even GBert models are an extension of the odd models, trained and evaluated with additional epochs because in them only the classifier part of the architecture is trained, excluding the BERT core.

Model	Epochs	Optimizer	Loss Function	Learning Rate
HistBert1	3	Adam	Focal Loss	1e-6
HistBert2	3	Adam	Cross Entropy	2e-5
HistBert3	3	AdamW	Focal Loss	1e-6
HistBert4	3	AdamW	Cross Entropy	2e-5
GBert1	2	Adam	Cross Entropy	2e-5
GBert2 (from GB1)	2	Adam	Cross Entropy	1e-5
GBert3	2	AdamW	Cross Entropy	2e-5
GBert4 (From GB3)	2	AdamW	Cross Entropy	1e-5
GBert5	3	Adam	Focal Loss	2e-5
GBert6 (from GB5)	2	Adam	Focal Loss	1e-5
GBert7	3	Adam	Cross Entropy	3e-5
GBert8 (from GB7)	2	Adam	Cross Entropy	1e-5
LYM	30	Adam	Focal Loss	1e-6

Table 9: Models configuration. The different configurations allow us to understand which are the optimal values of the hyperparameters for this metaphor detection task. The even GBert models (2,4,6,8) are an extension of the previous models, freezing the learning of the BERT core through the command *requires_grad=False* to consider only the classifier part. In the focal loss models, we used the same γ equals to 2.

5.2 Last year model

We tried to reproduce the results obtained by students from last year, calling their model *Last Year Model* or *LYM*. Last year's architecture is similar to our proposed model shown in Figure 4. They used a *HistBert Head* with 2 linear layers before the ReLU. They consider a *softmax* over three classes, so for each sentence, they assign a membership value to each class between 0 and 2 (0 for metaphors, 1 for candidates and 2 for non-metaphors). In the end, they have a 3-element vector for each sentence in which each element contains the value of belonging to the corresponding class. Hence, they assign the label using the *argmax* function. So every sentence corresponds to a single vector as the example:

True label	Predicted label		
	Metaphor	Candidate	Non-metaphor
Metaphor	1	0	0
Non-metaphor	0	0	1

Table 10: Example of LYM output data. For every element, we have a 1×3 vector filled with binary values.

We have not been able to reproduce the same results as previous students, due to the wrong implementation of the F1-score function and how the dataset splits are used. Considering the data as shown in the example 10, they compute the average of the binary values that are in column 0, column 1 and column 2. So they don't compute an F1-score based on precision and Recall of the model considering all three classes, but only on average between F1-score of 0 and F1-score of 1 (binary labels) for each class. Computing the correct F1-score, we obtain on the model provided as is a lower score (about $\frac{2}{3}$ lower). Another issue is that they use, to validate the model during training, the same sub-set used in the testing phase, the test subset, completely ignoring the validation subset they made.

Trying to reproduce the code with the correct subsets, we obtained an F1-score for metaphors of 0.12, even with a very low loss function value computed with binary cross-entropy in the *Focal Loss* function. In Table 11, we report the results obtained with their F1-score function, the results obtained with the correct per class F1-score over the unseen dataset and the results obtained from training and testing the correct model with our version of the dataset.

Method	Non-metaphors	Metaphors	Candidates
Code as is	0.83	0.53	0.74
Correct	0.84	0.12	0.68
Correct*	0.96	0.13	0.60

Table 11: Results, F1-score, of LYM model as is, over unseen data whit the correct F1-score function and correct but trained/tested with our version of the dataset (Correct*).

5.3 Results

After all the experiments, we obtain the best results with the *GBert8* model for the metaphors detection task, as shown in Table 12. As metrics, we used F1-score for each class, F1-score with average *Macro* and the accuracy as shown in Table 13. Every metric is an average computed over the 10 folds of cross-validation. The confusion matrix of GBert8 is in Figure 6.

Looking at the results, we can assume that our researches and experiments are a good start to obtaining some acceptable and encouraging results. We reached an F1-score of **0.31** with an Accuracy of **0.49** and a Recall of **0.47**, meaning our model manages to identify nearly 50 percent of the metaphors correctly. Which represents a good step forward compared to the architectures and models previously proposed.

Model	F1-score class 0	F1-score class 1	F1-score class 2	Macro F1
HistBert1	0.79	0.05	0.12	0.32
HistBert2	0.96	0.12	0.61	0.56
HistBert3	0.96	0.09	0.56	0.54
HistBert4	0.96	0.20	0.51	0.56
GBert1	0.96	0.08	0.58	0.54
GBert2	0.96	0.11	0.61	0.56
GBert3	0.95	0.16	0.34	0.48
GBert4	0.97	0.16	0.64	0.59
GBert5	0.93	0.00	0.00	0.31
GBert6	0.72	0.02	0.19	0.30
GBert7	0.95	0.30	0.35	0.53
GBert8	0.96	**0.31**	0.44	0.57

Table 12: Results of the experiments that correspond to the averages of per class F1-scores computed in each fold and the Macro averages as metrics. The model GBert8 obtained 0.31 as F1-score for the metaphors class (class 1). The configuration is shown in Table 9. The testing set is the same for every model as described in Section 3.1.

Model	Accuracy class 0	Accuracy class 1	Accuracy class 2
HistBert1	0.72	0.20	0.12
HistBert2	0.98	0.09	0.59
HistBert3	0.98	0.09	0.50
HistBert4	0.98	0.18	0.48
GBert1	0.99	0.06	0.49
GBert2	0.97	0.09	0.62
GBert3	0.99	0.18	0.24
GBert4	0.97	0.12	0.68
GBert5	1.00	0.00	0.00
GBert6	0.61	0.03	0.42
GBert7	0.97	0.43	0.26
GBert8	0.95	**0.49**	0.40

Table 13: Results of experiments: Accuracy. Different configurations are shown in Table 9. As the F1-score, every model is evaluated over the same test set and these values are an average over the 10 folds.

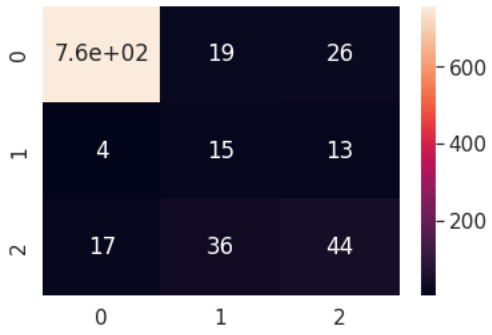

Figure 6: Confusion matrix of GBert8. In the diagonal we have the correct matches over non-metaphors (0), metaphors (1) and candidates (2).

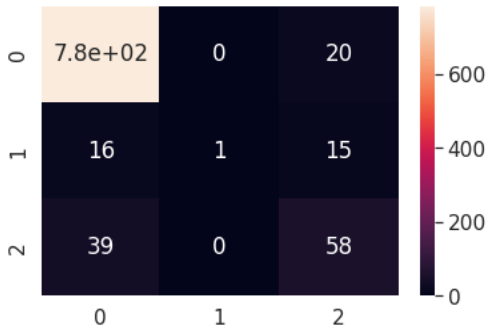

Figure 7: Confusion matrix of *GBert (NoAug)* trained on original data without data augmentation. The three classes are: *non-metaphors* (0), *metaphors* (1) and *candidates* (2).

All these results are an average over the 10 folds of the cross-validation.

Compared to the previous students model, as shown in Table 14, we can say that this GBert8 performs **better** in terms of *F1-score per class* and *Macro average*, so our model can recognize correctly more metaphors than the last year model (*LYM*).

6 Ablation study

The first step was data augmentation by Back translation on metaphors and candidates in order to train our model with a significant amount of relevant data. As shown in Tables 15 and 16 and considering the confusion matrix in Figure 7, the model *GBert (NoAug)*, even if it was trained following the same steps used for the model GBert8, produced slim results.

Model	F1-score class 0	F1-score class 1	F1-score class 2	Macro F1
GBert (NoAug)	0.95	0.05	0.61	0.54

Table 15: F1-score of *GBert (NoAug)* trained on original data without data augmentation.

Model	Accuracy class 0	Accuracy class 1	Accuracy class 2
GBert (NoAug)	0.97	0.03	0.61

Table 16: Accuracy of *GBert (NoAug)* trained on original data without data augmentation.

Comparing these scores with the scores obtained using the best model *GBert8* in Tables 12 and 13 and the confusion matrix in Figure 6, we can see that the data augmentation allow us to increase scores on the more relevant classes.

Considering the results obtained in (Devlin et al., 2018a) and the configuration of (Branden Chan, 2022), another problem was about finding an *optimal learning rate*.

For the BERT core the optimal learning rates to fine-tune it, as suggested in (Devlin et al., 2018a), are $5e{-}5$, $4e{-}5$, $3e{-}5$ and $2e{-}5$, depending on the training set size. Since our training set contains less than 8000 text fragments or sentences, it is defined as a small set and the recommended values are $2e-5$ and $3e-5$.

For the classifier, we followed the *hyperparameters optimization* technique, so we start with a higher value (0.001) decreasing this value in order to obtain better results. To produce the different models we used two different learning rates, $1e-6$ and $1e-5$, depending on which loss function is used, because with the *GBert6* that uses a learning rate equals to $1e-5$ in the Focal Loss function, we have found out that it generates the so-called *dying ReLU problem*. The *dying ReLU problem* (Lu et al., 2019) or *dead ReLU* is an event in which the ReLU layer always outputs the same value for any input, which means that it doesn't take part in discriminating between inputs. Once a ReLU ends up in this state it is unlikely to recover, because the function gradient at 0 is also 0, so gradient descent learning will not alter the weights.

Another aspect that we considered to produce an optimal model for this metaphors classification problem is the optimizer choice between the already introduced *Adam* and *AdamW*. We have practically tested the different results that weight decay, applied in these two different algorithms, can produce. As shown in Figure 8, with a small number of epochs the weight decay doesn't affect too much the results.

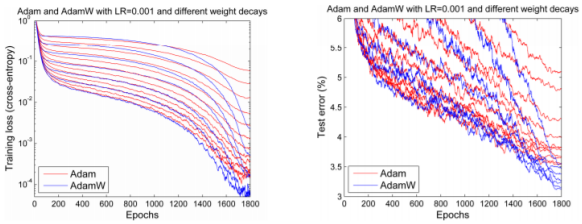

Figure 8: Adam vs AdamW. Figure from Fast AI.

The last part of our ablation study is referred to the analysis of the results of our best model. In Table 17 are shown the F1-scores obtained from each fold.

Fold	1	2	3	4	5	6	7	8	9	10
F1-score	0.25	0.50	0.31	0.36	0.00	0.25	0.22	0.00	0.44	0.40

Table 17: The F1-scores on the metaphor class obtained in each fold by the GBert8 model. The result shown in Tables 12 and 14 represents the average computed over these mid-results.

Model	F1-score class 0	F1-score class 1	F1-score class 2	Macro F1
*HistBert4	0.96	0.20	0.51	0.56
GBert6	0.72	0.02	0.19	0.30
GBert8	0.96	**0.31**	0.44	**0.57**
*LYM	0.96	0.13	**0.60**	0.54

Table 14: The comparison between our models and the previous year's model. The models shown in this table represent the best models considering the different configurations we have produced and tested. The *HistBert4* model uses the same HistBERT core as LYM (both are marked with * before the name) but uses the AdamW optimizer and cross-entropy loss. The *GBert6* model, which uses the Adam optimizer and focal loss, performs worse than the other models. The *LYM* performs better than our models in the candidate class, but poorly in the metaphor class which is the main one. Our best model *GBert8* performs well considering the main class and the average Macro. These results correspond to the averages of the F1 scores per class calculated in each fold of the cross-validation. The between-fold variance is calculated only for our best model (GBert8) and is shown and described in Section 6.

As we can see, folds 5 and 8 obtained **0.0** of F1-score for the metaphor class, so our model failed to classify even one metaphor of that fold correctly. This may be because there are more complex and less complex metaphors, given that some metaphors have led also human annotators to have problems in their classification. Even if we get a low **variance** ($\sigma = 0.026$) over the scores, we cannot say that our model performs well because it is able to correctly classify less than 50% of samples from the metaphor class.

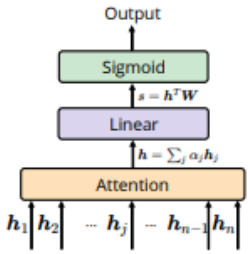

Figure 9: Attention for text classification. The idea is to use an attention mechanism between the head and the classifier. Figure from (Sun and Lu, 2020b).

7 Discussion

We started this experiment trying to reproduce the results obtained from last year's student project. We didn't achieve this goal easily, due to several problems described in Section 5.2, but from the beginning, their approach did not convince us and prompted us to try to understand the reasons for their choices. Hence the first part of this work was exclusively research and testing of other models and experiments, trying to understand what could be our starting point. After that, we started with our experiments, applying every piece of information that we learned. We think that from this work, one can easily use our model as a baseline or backbone and try it with different configurations.

We think our model is on the right track to solve this classification problem. Unfortunately, we didn't get high enough results. To achieve better results we need a more complex and structured architecture, that is able to understand different output representations. We already know that our output representation is too simple conceptually and sometimes complicated practically and this is the main reason for our results. We find that the representation proposed by the last year's students is interesting, but it needs to be well implemented.

Future works may include finding the elements that are more *important* in a text fragment to accomplish the task. That can be done by using an *attention mechanism* in order to force the model to pay more attention to the most relevant elements. An attention mechanism on text classification (Dzmitry Bahdanau and Bengio, 2015) (Jain and Wallace, 2019) is focused on finding which tokens are most relevant to the specific task. Following the work done in (Sun and Lu, 2020a), future works may consider using *attention scores* as a global measurement of the significance of word tokens. This attention part could be the missing piece to achieve this task and we think that using an attention mechanism between the head and the classifier, as shown in Figure 9, we might obtain better results.

Another future improvement that can be done is considering an *Optimal Transport* (Bhardwaj et al., 2022) algorithm to compute the distances between the words of a sentence, in order to obtain a value and to compute a threshold that allows us to easily recognize if a text contains at least

one metaphor or not. Compute this distance is perfectly in line with the context breach approach and it represents a great challenge.

Another approach to solve the small dataset problem, besides creating a bigger labeled dataset, may be to use a reinforcement learning approach like in (Schneider et al., 2022). This method might be considered to train new models on larger quantities of non-labeled data, using then the existing dataset only for testing.

8 Conclusion

At the beginning of this experiment, our goal was to find and set an optimal starting point and *baseline* to potentially solve the *German metaphors classification problem* with *BERT*. We achieved this goal by producing an interpretable and ready-to-use dataset to train/test the model, and a simple but working architecture that can understand a little bit of what a metaphor is, obtaining an average of **0.31** on the F1-scores for the metaphor class computed over 10 folds as shown in Table 17. We produced a model that is able to recognize some metaphors and that can be used as a baseline for other future works. Our approach was not focused on obtaining the highest scores, even if we get it compared to previous years, but on building something understandable, that works and that can be reproduced.

In general, the model performs not well as we want, even if BERT is a powerful language model that has achieved state-of-the-art performance on various NLP tasks, including some semantic tasks, it still faces several challenges in metaphor detection. One of the main difficulties of BERT in metaphor detection is that metaphors are often indirect and implicit, making them hard to detect using simple pattern matching or word co-occurrence techniques. Metaphors often rely on complex and abstract concepts that can be challenging to encode in a vector representation. Moreover, BERT models typically rely on pre-defined tokenization schemes that split the text into individual words or subwords, which can be problematic for metaphors that involve multiple words or phrases that are semantically connected. Even if BERT models can capture some aspects of semantic relationships between words, they may not be able to fully capture the subtle nuances of metaphorical language. Metaphors often involve creative or poetic language use, and may not conform to the conventional linguistic patterns captured by BERT's training data. Overall, while BERT can be a useful tool for metaphor detection, it still faces several challenges in accurately detecting and interpreting metaphorical language in natural text.

References

Rishabh Bhardwaj, Tushar Vaidya, and Soujanya Poria. 2022. Towards solving nlp tasks with optimal transport loss. *Journal of King Saud University - Computer and Information Sciences*, 34(10, Part B):10434–10443.

Malte Pietsch Tanay Soni Branden Chan, Timo Möller. 2022. Deepset.ai bert model. `https://huggingface.co/bert-base-german-cased?doi=true`.

Minjin Choi, Sunkyung Lee, Eunseong Choi, Heesoo Park, Junhyuk Lee, Dongwon Lee, and Jongwuk Lee. 2021. Melbert: Metaphor detection via contextualized late interaction using metaphorical identification theories. *arXiv preprint arXiv:2104.13615*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018a. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018b. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Erik-Lân Do Dinh and Iryna Gurevych. 2016. Token-level metaphor detection using neural networks. In *Proceedings of the Fourth Workshop on Metaphor in NLP*, pages 28–33.

Kyunghyun Cho Dzmitry Bahdanau and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *ICLR*.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *CoRR*, abs/1508.01991.

Sarthak Jain and Byron C. Wallace. 2019. Attention is not explanation. *CoRR*, abs/1902.10186.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

George Lakoff and Mark Johnson. 1980. Conceptual metaphor in everyday language. *The journal of Philosophy*, 77(8):453–486.

Yucheng Li, Shun Wang, Chenghua Lin, Frank Guerin, and Loïc Barrault. 2023. Framebert: Conceptual metaphor detection with frame embedding learning. *arXiv preprint arXiv:2302.04834*.

Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. *CoRR*, abs/1708.02002.

Jerry Liu, Nathan O'Hara, Alexander Rubin, Rachel Draelos, and Cynthia Rudin. 2020. Metaphor detection using contextual word embeddings from transformers. In *Proceedings of the Second Workshop on Figurative Language Processing*, pages 250–255.

Ilya Loshchilov and Frank Hutter. 2017. Fixing weight decay regularization in adam. *CoRR*, abs/1711.05101.

Lu Lu, Yeonjong Shin, Yanhui Su, and George Em Karniadakis. 2019. Dying relu and initialization: Theory and numerical examples. *arXiv preprint arXiv:1903.06733*.

Felix Schneider, Sven Sickert, Phillip Brandes, Sophie Marshall, and Joachim Denzler. 2022. Metaphor detection for low resource languages: From zero-shot to few-shot learning in middle high german. In *Proceedings of the 18th Workshop on Multiword Expressions@ LREC2022*, pages 75–80.

Xiaobing Sun and Wei Lu. 2020a. Understanding attention for text classification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3418–3428.

Xiaobing Sun and Wei Lu. 2020b. Understanding attention for text classification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3418–3428.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

Appendix

Work distribution

Sergio Solmonte - 4282289

- Abstract;
- BERT introduction;
- Back translation;
- Architecture;
- Data: examples;
- Data augmentation;
- Loss functions;
- Cross Validation;
- GBert Models (Experiments);
- Ablation study: Not augmented data and dying ReLU;
- GitHub Repository.

Marco Madonna - 4282300

- Introduction;
- Related works;
- German Bert and Historical Bert;
- Data;
- Optimizers;
- HistBert models and LYM (Experiments);
- Ablation study: learning rates and optimizers.