Natural Language Processing

Assignment 1

Rajbir Bhattacharjee

R00195734

Contents

[Review of Current Approaches 2](#_Toc88849002)

[GermEval2021 3](#_Toc88849003)

[Other Sources 4](#_Toc88849004)

[Compound Words (Komposita) 5](#_Toc88849005)

[Sarcasm Detection 5](#_Toc88849006)

[Discussion 6](#_Toc88849007)

[Strategy to add more datasets 7](#_Toc88849008)

[Strategy 7](#_Toc88849009)

[Implementation 7](#_Toc88849010)

[Data Augmentation 8](#_Toc88849011)

[Pre-Processing Pipeline 9](#_Toc88849012)

[Cleaning with Python 9](#_Toc88849013)

[Cleaning with Spacy 9](#_Toc88849014)

[Parts of Speech Tagging and Named Entity Recognition 10](#_Toc88849015)

[Additional Features 10](#_Toc88849016)

[Vectorization 10](#_Toc88849017)

[Handling imbalance 10](#_Toc88849018)

[Traditional ML Classifiers (including Naïve Bayes) 11](#_Toc88849019)

[Use of n-Grams 12](#_Toc88849020)

[Fine Tuning Classifiers 12](#_Toc88849021)

[Results From Fine Tuned Classifiers 13](#_Toc88849022)

[Accuracy 13](#_Toc88849023)

[F1-Score 13](#_Toc88849024)

[Language Model – Transformers 13](#_Toc88849025)

[Transfer Learning 16](#_Toc88849026)

[Transformers in Transfer Learning 17](#_Toc88849027)

# Review of Current Approaches

In this literature review, we looked both papers from GermEval 2021 and elsewhere. We did not constrain our review to papers on German language alone, but extended that to other languages. We will first start with the reviewing approaches taken at GermEval2021 and then review approaches elsewhere.

## GermEval2021

In GermEval2021, a hand curated small dataset of online German comments was created, and the task was to categorize it under three categories: toxic comments, fact claiming, and engaging

The majority of the participants used transformers along with other techniques. While some teams in GermEval2021 reported benefits of multi-task learning, other teams reported that there was no benefit in the tasks. Also, the addition of synthetic data, and augmenting data from other sources had mixed results.

Calizzano et. al[[1]](#endnote-1) used XLM-RoBERTa and MT5, both of which are transformers. They pre-trained the transformer on the dataset and that resulted in a 10% improvement in F1-score. They also used data augmentation via two different methods: (1) training on other multi-lingual datasets that include German comments and (2) created a dataset of Facebook comments by classification with a model and keeping the comments classified as toxic or non toxic with a probability of larger than 0.8. However, they found that data augmentation was not very helpful, and in some cases, it actually reduced the accuracy. This could, however, be due to the fact that their method of augmentation relied on another classification network. They reported F1-scores in the ball park of 0.68

Schaefer and Stede [[2]](#endnote-2)used a used BERT encodings and classified them using three techniques: (1) SVM, (2) XGBoost and (3) another Transformer-based classification network. They found that BERT+SVM performed better on the test-set but Bert+Transformers performed better on the evaluation set. They report an F1-score of between 0.69 and 0.736.

Hilderbrandt et al. used ensembles of Logistic Regression and SVM classifiers. For feature extraction, they used a combination of German BERT encodings from a huggingface model and AdHominem[[3]](#endnote-3). They report F1-scores between 0.66 and 0.72.

Tran and Kruschwitz[[4]](#endnote-4) use BERT encodings from XLM-RoBERTa and use an ensemble classifier with majority voting. They also try to translate German text to English and then apply transformers to it before ensembling with majority voting. They also use several BERT based models for classification which are then ensembled by majority voting. Overall, they report F1-scores between 0.68 and 0.76.

Böck et al[[5]](#endnote-5) use a compare mBERT, XLM-R and GottBERT in the task of fact claiming detection and find that GottBERT performs the best. They augment the dataset by using the ClaimBuster dataset. They also augment the dataset further by translating other datasets from English to German. Overall, they report F1-scores in the ball park of 0.74.

Gémes and Recski[[6]](#endnote-6) use Rule-based and hybrid methods for detecting Toxic, engaging and fact-claiming comments. They find that for toxicity detection, if a sentence contains at least two words with at least 4 characters each written in caps, a rule to detect this achieved 91% precision and 4% recall on its own on the validation set. They searched for patterns in toxic comments with unigrams, bigrams of words and lemmas with and without POS tagging for disambiguation. By carefully hand-crafting rules, they were able to achieve F1-scores between 0.79 and 0.82.

Schmidhuber[[7]](#endnote-7) approached GermEval2021 by attempting to generate synthetic data to augment the given data, and classify it using mBERT. It was found that synthetic data actually reduced the performance of the system. Overall F1-scores of 0.618 and 0.615 were reported for the model without and with synthetic data.

Schütz[[8]](#endnote-8) et al. performed toxic comment detection by using BERT and augmenting the data with a set of 1 million tweets. They further enriched BERT encodings with hand-crafted features like word-count, punctuation-count, exclamation-count, questionmark-count, ratios between punctuations, emoji-count, word-emoji ratio etc. They also counted the number of hate-words and hate word count ratio using a list of hate words. They also used a dictionary of words to detect the “sentiment” feature. They were able to achieve an F1-score of 0.74 on the validation data.

Haak and Engelmann[[9]](#endnote-9) classify for toxicity by extracting features and classifying with Logistic Regression, Support Vector Classification, Linear SVC and multi-layer perceptrons. Their features account for smileys, questions, emojis, caps, short sentences and ellipses. They also Snorkel[[10]](#endnote-10) to determine coverage, overlps, conflicts and empirical accuracy. They find that SVC and Linear Regression perform the best, however, they report F1-scores in the range of 0.51 to 0.57 only. Panda and Levitan[[11]](#endnote-11) generated embeddings using transformers and then experimented with logistic regression and linear classifiers. They found that multi-task learning did not improve the performance of the classifiers. Akomeah et al.[[12]](#endnote-12) build on the work by Hoffman and Kruschtitz[[13]](#endnote-13) used transformer-based encodings and LSTMs and then used these with either SVM or ANN classifiers which were finally ensembled.

Arjun et al.[[14]](#endnote-14) use the embeddings obtained by different methods, and using those embeddings with a classifier to perform all three tasks. They try both embeddings obtained from both Glove and Transformers. As the classifier, they try four different approaches: CNNs, Capsule Networks, RNN/GRU and Ensembles. They also enrich the encodings by hand-crafted features like number of words, urls, punctuations, all-cap words, positive and negative words, moderator mentions and distribution of emojis. They find that ensembles perform the best. They report F1-scores in the range of 0.66 and 0.74.

Gawron and Schmidt[[15]](#endnote-15) compared nine pre-trained transformers and compared the performance, and found that GELECTRA and GBERT perform the best. They also tried to add data from previous GermEvals but that reduced their overall accuracy and F-score.

Morgan et al.[[16]](#endnote-16) used transformer based classifiers and found that multi-task learning yields better results than single-task learning. This may be because the tasks of toxicity detection, fact-claiming detection and engagement detection may rely on similar underlying features, and thus may benefit from multi-task learning. Chan et al.[[17]](#endnote-17) find that pre-training transformers with a larger corpus of similar text can yield better results. They specifically pre-train on the following datasets: OSCAR, OPUS, Wikipedia and OpenLegalData.

Bornheim et al.[[18]](#endnote-18) compared several transformers for all three tasks. They also experimented with different activation functions. They also tried ensemble methods and both mult-label and single-label classifiers. They found that none of these made a significant difference.

## Other Sources

We will now review some approaches by sources other than the participants of GermEval2021.

Risch and Krestel[[19]](#endnote-19) explored ensembling Logistic Regression, GRU’s and Syntactic features with Gradient Boosted Decision trees to detect aggression in online text. The same authors also explored bagging the output of BERT based models.[[20]](#endnote-20) In another work, the same authors use up-votes and down-votes to detect engaging comments.[[21]](#endnote-21)

Malmasi and Zampieri[[22]](#endnote-22) have found that it is difficult to distinguish between hate speech and profanity. In a different work, Zampieiri et al.[[23]](#endnote-23) found that Bi-LSTM with CNNs outperformed SVM based models and Bi-LSTMs alone in the task of offensive speech detection.

Ranasinghe et al.[[24]](#endnote-24) compared 7 different deep neural-network approaches (Pooled GRU, Stacked LSTM with attention, LSTM and GRU with attention, 2D convolution with pooling, GRU with Capsule Networks, LSTM with Capsule and Attention, and BERT), and found that BERT based networks performed the best on aggression detection.

Zhao et al.[[25]](#endnote-25) compare the performance of downstream classifiers after applying BERT to the task of toxic comment detection. They find that simple linear neural networks with BERT outperform more complex neural networks (like Bi-LSTM and CNN) with BERT. D’Sa et al.[[26]](#endnote-26) found that BERT models with fine-tuning outperformed FastText when they were combined with simple deep neural networks.

Maslej-Krešňáková[[27]](#endnote-27) did an in-depth comparison of several methods of pre-processing with popular transformer models. They also compare transformer based models with more traditional language models like Bi-LSTM with similar preprocessing techniques.

Safaya et al.[[28]](#endnote-28) find that BERT when combined with CNN performs better than BERT alone when trying to detect offensive speech in Arabic.

Song et al.[[29]](#endnote-29) found that model-fusion of several transformers with a fusion of BCE-loss and Focal-loss functions performs best at toxic text detection when there is data imbalance.

Hu et al.[[30]](#endnote-30) discover that while models trained on English approach human levels on many classification tasks, models trained on cross-lingual datasets lag behind.

## Compound Words (Komposita)

Both English and German have compound words. In English compound words are separated by a hyphen, for example, father-in-law is one such word. In german, the hyphen is not used. For example, ‘hundemüde,’ in German means dog-tired.

In NLP tasks, sometimes splitting compound words can help. In the case of sentiment analysis and toxic comment detection, this might be most useful.

Marco[[31]](#endnote-31) proposed a frequency based approach with form-to-lemma mappings to split compound words in German. Their method is an extension of the work by Koehn and Knight[[32]](#endnote-32). Given many possible splits of a word, Koehn and Knight pick up the split with the highest geometric mean of frequencies of the individual words of the split occurring in a given corpus. Koehn and Knight’s algorithm is tuned for translation tasks where it is possible to refer to the corpus of another language. Bay et al.[[33]](#endnote-33) have used abbreviation detection via TF/IDF and FastText followed by compound splitting with FastText and found this helpful in processing medical text. Krotova et al.[[34]](#endnote-34) use a deep learning based approach to splitting German compound words. Ma et. al.[[35]](#endnote-35) use a Conditional Random Field model to predict a label for each letter in a compound word to indicate whether the letter is the beginning of a word (B), end of a word (E), an intermediate letter of a word (M), or a letter in a single-letter word (S). The sequence of B-M-E-S labels can then be used to split the word.

However, we were unable to find any library which performed this task reliably, and hence compound word splitting was not done in this assignment.

## Sarcasm Detection

Detection of Sarcasm may be key to sentiment and emotion detection. Chauhan et al.[[36]](#endnote-36) manually annotated the MUStARD dataset with both sentiment and emotion classes, and then proposed a novel attention mechanism that can perform well on multi-task learning that involves sentiment and emotion analysis, sarcasm detection. Barnes et al.[[37]](#endnote-37) deem that detection negation is necessary for sentiment analysis, and they find that multi-task learning performs better at detection of negation. They use a combination of a Bi-LSTM network along with a CRF-tagger.

## Discussion

In all the papers reviewed here, some tasks were commonly performed as pre-processing tasks before actual classification – these tasks include tokenization, stop-word removal, removal of punctuations, parts-of-speech tagging, lemmatization/stemming, hyperlink removal, conversion of smileys to text, removal of named entities, replacement of abbreviations with full forms, replacement of numbers with a specific text to indicate the presence of a number, removal of symbols etc.

The above tasks can be performed by a variety of methods. Some of these are performed using regular expressions, others are performed using trained models.

Following these tasks, the next task was to convert the words to a vector representation. Different methods were used here – in some of the work, TF/IDF, count vectorization or one-hot encoding was used. In others, word-to-vec or Glove like representations were used. A lot the work cited above used transformers for the task of vectorization.

There are several advantages of using transformers over more traditional sequence models like LSTM:

1. LSTM etc. are hard to train because of deep recurrent nature
2. Transformers are naturally more conducive to being paired with a secondary classifier because a transformer can produce a fixed length vector that describes the document that is passed through it.
3. The advantage of LSTM is that it can take a variable length input and then produce a fixed length output. While this is a big improvement over other neural networks like CNNs and feed forward networks, transformers can do even better. A transformer can be used to produce a positional encoding of the entire document that is of a fixed length. The transformer includes information about the relative positions of the words in relation to each other by incorporating a Fourier transform, and subsequent classifiers can classify using that.
4. While transfer learning with LSTM is possible, it has traditionally not worked very well for most applications, and most LSTM networks need to be trained from a scratch. On the contrary, large pre-trained transformer networks have proven to do very well on a wide variety of tasks.
5. Recurrent neural networks are limited in parallelism because of the nature of the recurrence. Transformers can offer more parallelism because there is no recurrence. This can be very useful as modern GPUs are capable of high parallelism.

Given the above, it is not surprising that transformers have been a favorite amongst all researchers. Transformers can be used in several ways, before using a classifier:

1. Either use only the vectorizer from a commercially available transformer package, like BERT, and use a secondary classifier
2. Or use the final output from a transformer layer with a secondary classifier
3. Compose a new neural network with a pre-trained transformer and fine tune to the task of prediction

Some of the approaches described above performed enrichment with other other hand-crafted features. Some interesting features were:

1. Number of words with all-caps
2. Specific attention to negative and positive emojis
3. Use of dictionary to include negative and positive words

A host of the approaches described above enriched the dataset with secondary datasets. Some interesting ways in this was done are listed below:

1. There is an abundance of datasets in English. Some researchers translated those datasets to English.
2. Some researchers used datasets where shared learning is possible with similar target classes.
3. Some researchers used mult-lingual datasets available on similar tasks that also include German text
4. One researcher simply translated the German text into English and then used a classifier trained to recognize English labels.
5. Some approaches generated synthetic data

Overall, the effect of enriching the data or trying to create synthetic data was mixed, and there was no clear trend in the results.

# Strategy to add more datasets

## Strategy

Some ways in which datasets can be added are:

1. Add other German datasets on the task of sentiment analysis, hate speech detection etc. For example:
   1. The GermEval 2019 dataset for identification of offensive language. There can be some correlation between offensive language and toxic speech, but as Malmasi and Zampieri[[38]](#endnote-38) have pointed out, it may be difficult to distinguish between hate speech and profanity. However, since the corpus is the same, pre-training with the datasets can still be used as an effective transfer learning method.
   2. The GermEval 2018 dataset is more aligned to the GermEval2021 task. In 2019, the dataset was to make two shared predictions: toxicity classified into 6 different categorical non-mutually exclusive levels (toxic, severe toxic, ob-scene, insult, identity hate and threat) and identification of aggression on a mult-lingual dataset. With some re-mapping of labels, this dataset can be used as well.
   3. The GermEval 2017[[39]](#endnote-39) dataset deals with polarity detection and opinion detection. This is not relevant to the tasks we have at hand here, but the datasets can be used for pretraining and transfer learning nonetheless.
2. There are several English datasets available on the three tasks. Inspired by the approaches discussed above, we can take those datasets and use machine translation to form a German dataset for the same.

## Implementation

The GermEval2018 task was similar to the first task of detecting toxic comments. In the GermEval2018 task, German comments were given one or more tags, which included:

1. Insult
2. Offence
3. Profanity
4. Abuse
5. Other

It was assumed that the presence of the categories insult, offence or abuse is equivalent to toxicity. The dataset was relabeled and appended to the GermEval2021 dataset only for the task of toxicity detection. The re-labeled 2018 dataset can be found here:

<https://raw.githubusercontent.com/bhattacharjee/mtu-nlp-assignment/main/assignment1/germeval2018_a.txt>

When this additional dataset was added, it gave a significant improvement in performance.

# Data Augmentation

Data Augmentation can be an important way to improve the accuracy of any machine learning system, especially when there is little training data available.

Some classic data augmentation techniques for NLP tasks are:

1. Random deletion. In some documents, some words are randomly deleted.
2. Random insertion. In some documents, some words are inserted randomly.
3. Random swap. Two words a are chosen at random and their positions are swapped
4. Synonym replacement. Randomly chosen words are replaced with their synonyms from a dictionary.
5. Albumentation methods, these include:
   1. Shuffling sentences or sequences in the document
   2. Removing duplicate sentences or sequences in the document

In addition some other methods can be as follows:

1. Using a dictionary that has the sentiment, add smileys and emojis based on the sentiment
2. Convert smileys and emojis to their respective words
3. Convert well known abbreviations to their full forms

Feng et al.[[40]](#endnote-40) did an extensive survey of data augmentation approaches for NLP. They find that most approaches either create new data by modifying the existing data or take a different approach to create synthetic data. The goal most approaches take is to have the augmented data to act as a regularization step. In the same paper they note that the approach by Boyd[[41]](#endnote-41) in which he uses grammatical error correction for German text can also be used as an augmentation technique.

Feng et al. further classify augmentation techniques into three types:

1. Rule-based – this includes random deletion, insertion and swap, and dependency tree morphing.[[42]](#endnote-42)
2. Example interpolation based – this includes mixed sample data augmentation and MixUp where pairs of inputs are fused together.[[43]](#endnote-43) [[44]](#endnote-44)
3. Model-based – this includes backtranslation methods where an example from one language is translated into another language and then back into the same language again and using RNNs/Transformers etc to generate examples. Ng et al. propose corrupt and reconstruct methodology where a word or a group of words is masked and then the string is reconstructed with transformers. [[45]](#endnote-45)

Feng et al. note that there are several uses of data augmentation:

1. Generating examples in low resource languages
2. Handling class imbalance. For fixing class imbalance SMOTE can also be applied.
3. Few shot learning
4. Mitigating bias – replacements of biased terms can help here, for example replacing he with she, and male names with female names etc.

# Pre-Processing Pipeline

## Cleaning with Python

The data was first loaded using pandas. After that, regular expressions were used to perform the following:

1. Convert to lowercase
2. Emojis were replaced with their descriptions. Certain emojis can be relevant to the tasks at hand. The descriptions were modified to be a single word separated by underscores, eg. \_\_thumbs\_down\_\_. These emojis are not German, but that should not make any difference to the models.
3. The roles like @user, @moderator, etc. was removed. This was done because it was assumed that this might introduce bias into the classification, although the description of. The dataset says that chances of a bias are very unlikely.
4. Ellipses are removed
5. Any numbers are replaced with a tag, like NUM
6. URL’s and links are removed
7. Remove any punctuations
8. Punctuations at the beginning or end of words are removed

## Cleaning with Spacy

After the above set of steps, further cleaning was performed via a dedicated NLP toolkit. NLTK and spacy were both evaluated, but spacy seemed to be a better library for some tasks, and this was chosen for all asks as a result. The following operations were performed with spacy.

1. Numbers or symbols are removed, we have already performed this step earlier, but some numbers may still be present.
2. Stopwords are removed
3. Punctuations are removed, again this was already done via regular expressions but some may still remain.
4. Words are lemmatized

## Parts of Speech Tagging and Named Entity Recognition

Experiments were tried with both POS tagging and removal of named entities, and without POS tagging and still having named entities. It was found that removal of named entities gave a big boost to model performance. Also, it was found that POS tagging gave a further small gain in model performance.

Both POS tagging and named entity removal were performed by use of the spacy library.

## Additional Features

Taking inspiration from the approaches taken by various teams in the GermEval2021 competition, the following features were added:

1. Number of words with length greater than 3 that have all letters in capital
2. Number of exclamations
3. Ratio of exclamations to number of characters

## Vectorization

The text-features need to be vectorized before they can be used with a classifier. Three experiments were done

1. CountVectorizer alone. This produced the worst results and hence this wasn’t explored further. The code submitted doesn’t have this.
2. Count Vectorizer along with TF-IDF. This performed better than word counts alone. The scheme is shown below.

Diagram

Description automatically generated with medium confidence

1. Count vectorizer, TF-IDF and term-frequences, all three simultaneously. This produced the best results by far. Final optimization was performed on this set. The scheme is shown below.

Diagram, table

Description automatically generated with medium confidence

## Handling imbalance

The dataset showed a small amount of imbalance. A few things were attempted to take care of the imbalance:

1. Models were tried with and without SMOTE, and it was fond that this did not have much of an effect on model performance.
2. With some classifiers, *a class balanced weighting was taken*, and it was found that this performed better than an even weighted classification.

Further work could involve the following:

1. Generating artificial samples by data augmentation
2. Experimenting with different cut-offs
3. Using the other variants of SMOTE

# Traditional ML Classifiers (including Naïve Bayes)

To get an initial feel of how the model performs, the following classifiers were tried for each of the three tasks:

1. Linear SVC
2. Multinomial Naïve Bayes
3. Bernoulli’s Naïve Bayes
4. Random Forest Classifier
5. Ada Boost
6. SVM Classifier with default settings
7. SVM Classifier with RBF kernel
8. Quadratic Discriminant Analysis

The results with the top 3 accuracies and top 3 F1-scores for each task are shown below:

================================================================================

Sub1\_Toxic

----------

classifier task\_name metric smote value

49 BernoulliNB\_nosmote Sub1\_Toxic accuracy 0 0.67209

1 LinearSVC\_nosmote Sub1\_Toxic accuracy 0 0.662078

31 MultinomialNB\_nosmote Sub1\_Toxic accuracy 0 0.659574

classifier task\_name metric smote value

2 LinearSVC\_nosmote Sub1\_Toxic f1\_score 0 0.470588

8 LinearSVC Sub1\_Toxic f1\_score 1 0.470149

56 Quadratic Discriminant Analysis Sub1\_Toxic f1\_score 0 0.463104

================================================================================

Sub2\_Engaging

-------------

classifier task\_name metric smote value

39 RandomForestClassifier Sub2\_Engaging accuracy 1 0.829787

45 RandomForestClassifier\_nosmote Sub2\_Engaging accuracy 0 0.828536

21 SVC Sub2\_Engaging accuracy 0 0.827284

classifier task\_name metric smote value

40 RandomForestClassifier Sub2\_Engaging f1\_score 1 0.624309

16 AdaBoost Sub2\_Engaging f1\_score 0 0.596774

4 LinearSVC\_nosmote Sub2\_Engaging f1\_score 0 0.583569

================================================================================

Sub3\_FactClaiming

-----------------

classifier task\_name metric smote value

47 RandomForestClassifier\_nosmote Sub3\_FactClaiming accuracy 0 0.755945

11 LinearSVC Sub3\_FactClaiming accuracy 1 0.744681

5 LinearSVC\_nosmote Sub3\_FactClaiming accuracy 0 0.740926

classifier task\_name metric smote value

36 MultinomialNB\_nosmote Sub3\_FactClaiming f1\_score 0 0.6

12 LinearSVC Sub3\_FactClaiming f1\_score 1 0.587045

6 LinearSVC\_nosmote Sub3\_FactClaiming f1\_score 0 0.578411

## Use of n-Grams

Typically, these vectorization techniques like TF-IDF and Count Vectorization do not include any information about the relative position of words. Often, the relative position of words is essential to discovering the meaning. For example, ‘Live to Work’ and ‘Work to Live’, mean completely different things, but models that rely only on TF-IDF or Count Vectorization will treat these as the same.

One way to overcome this is to use n-Grams, where groups of n-words are considered as the same words. So, in case of ‘Live to Work’, the n-Grams would be as follows:

{‘Live’, ‘To’, ‘Work’, ‘Live to’, ‘to Work’, ‘Live to Work’}

In Scikit learn, vectorizers take a useful parameter called ngram\_range, which can be used to specify the range of n in n-Grams.

This was experimented with, but it was discovered that in the case of the three tasks, including n-Grams led to worse accuracy than not using them. The reason for this is not known, but one reason could be that n-grams introduced too many individual words, which made the signal noisy.

In the results presented n-grams were not used.

## Fine Tuning Classifiers

The best models (using the best F1-score) from the previous notebook were taken and the best hyper-parameters searched with cross-fold validation. Also, the expanded dataset with GermEval2018 data was used for sub-task 1.

For Sub1\_Toxic the following scikit-learn pipeline was used:Diagram

Description automatically generated

For Sub2\_Engaging the following scikit-learn pipeline was used:

Graphical user interface, text, application

Description automatically generated

For Sub3\_FactClaiming the following scikit-learn pipeline was used:

Graphical user interface, diagram, text

Description automatically generated with medium confidence

## Results From Fine Tuned Classifiers

|  |  |  |  |
| --- | --- | --- | --- |
| Model | Task | Accuracy | F1-Score |
| Linear SVC | Sub1\_Toxic | 0.656 | 0.517 |
| Random Forest | Sub2\_Engaging | 0.812 | 0.618 |
| Multinomial Naïve Bayes | Sub3\_FactClaiming | 0.705 | 0.603 |

# Language Model – Transformers

Since a majority of the teams in GermEval2021 used transformers, one experiment with transformers was performed. Transformers from HuggingFace were used. These transformers were pre-trained on uncased German corpora. Regular cleaning was performed as in the case of traditional classifiers.

Two different networks were used

1. BERT with a Dense Neural Network

This model was trined for 5 epochs, and it failed to produce good results. The best validation accuracies after 5 epochs were as follows:

|  |  |
| --- | --- |
| Task | Accuracy |
| Sub1\_Toxic | 0.36 |
| Sub2\_Engaging | 0.29 |
| Sub3\_FactClaiming | 0.36 |

It is apparent that this is quite disappointing.

Diagram, table

Description automatically generated

A Deeper network was also tried (it is not submitted in the code), but that also didn’t produce very good results.

Table

Description automatically generated

1. BERT with a CNN network  
   This model failed to run as it hit an out-of-memory condition in google colab. It is unclear why this failed to run, and it was not investigated further.

Table

Description automatically generated

The results with BERT were disappointing, and it needs more research to find out why BERT underperformed so badly. One reason could be that there was so little data that the network could not train at all.

# Transfer Learning

Transfer learning has proven to be a big success in many fields, especially tasks that involve learning from visual data. The theory is that lower-level features learnt in one task can be applicable to other tasks as well. For example in image processing, similar tasks like identifying lines, gradients, colors, objects etc. are applicable to all image processing tasks. Therefore, a network that is specialized to do one task in image processing learns a lot of underlying features that may be helpful in other tasks as well.

The general way in which transfer learning is performed is to take a pre-trained network for task X, remove the last few layers of the network, and add new layers and retrain on task Y. The result is often better than training a system from scratch.

This has led to the development of pre-trained libraries by third parties – like MobileNet etc. Some of these pre-trained libraries are trained on large infrastructures over long periods of time that are not available to most developers. However, the advantage is that once these are trained on these large infrastructures, they can be reused with little effort.

It is easy to see that transfer learning will be useful in natural language processing as well. However, that remained an elusive goal for a long time. LSTM’s and RNN’s did not take to transfer learning too well, and generally each RNN had to be trained on the task specific to it from a scratch.

One of the first successful attempts at transfer learning for NLP was Howard and Ruder in ULMFit[[46]](#endnote-46). Howard and Ruder suggest that NLP models are typically shallower than computer vision methods, and require a different protocol for fine-tuning. They propose the following protocol for fine-tuning existing NLP models:

1. Discrminative fine-tuning. Instead of using the same learning rate for all layers, they propose using different learning rates for different layers
2. Slanted Traingle Learning Rate. They also propose changing the learning rate across the epochs in a triangular method. The learning rate is first linearly increased till a particular epoch, and then decreased linearly.
3. Concat Pooling. They propose that the crux of any large piece of text lies in only a few short sequences. To capture this, they propose concatenating the following: maxpool of the hidden features of all time stamps, minpool of the hidden features of all time-stamps, and the hidden feature at the current timestamp.
4. Gradual Unfreezing. Unfreezing all layers at one go runs the risk of catastrophic forgetting. Howard and Ruder suggest gradually unfreezing the layers. In this method, the last layer is unfrozen and then it is fine-tuned. After this, the last and the last-but-one layers are unfrozen and then they are fine-tuned. This is repeated till most layers have been trained.

## Transformers in Transfer Learning

Transformers were first proposed by Vasvani et al.[[47]](#endnote-47) and have become very popular in NLP tasks. There are many advantages of transformers, but some of the most important ones are:

1. While LSTMs need to be trained sequentially, one token at a time, transformers can look at an entire document with multiple tokens at the same time. This leads to greater parallelism and allows for training more complex networks on highly parallel machines.
2. The attention mechanism leads to greater accuracy.
3. Transformers use a positional encoding strategy by using sines and cosines, similar to Fourier transforms, that give summary information about the relative positions of the words. This is key to the parallelism while avoiding dealing with recurrent elements in the network.
4. The masking strategy in transformers ensures that Transformers learn to generalize better as certain words are masked out during the training process, so that the transformer cannot over-rely on some aspects. This has the knock-on effect of being able to transfer well to other tasks.

Transformers have been found to perform well with transfer learning, and are very popular currently.

There is also an abundance of libraries that make the task of transfer learning with transformers easier. FARM from deepset.ai is very popular.[[48]](#endnote-48) Future work should explore applying FARM to the GermEval2021 tasks.

# Use of Parsing Techniques

Two parsing techniques were used:

1. Named entities were recognized and removed. This gave a big boost in performance.
2. The POS tags were appended to the words, separated by a colon (:). This led to a small increase in performance.

For both these tasks, Spacy was used.

1. XLM-RoBERTa and MT5, both of which are transformers. They pre-trained the transformer on the dataset and that resulted in a 10% improvement in F1-score. They also used data augmentation via two different methods: (1) training on other multi-lingual datasets that include German comments and (2) created a dataset of Facebook comments by classification with a model and keeping the comments classified as toxic or non toxic with a probability of larger than 0.8. However, [↑](#endnote-ref-1)
2. Schaefer, R., & Stede, M. (2021, September). UPAppliedCL at GermEval 2021: Identifying Fact-Claiming and Engaging Facebook Comments Using Transformers. In *Proceedings of the GermEval 2021 Shared Task on the Identification of Toxic, Engaging, and Fact-Claiming Comments* (pp. 13-18). [↑](#endnote-ref-2)
3. Boenninghoff, B., Kolossa, D., & Nickel, R. M. (2021). Self-Calibrating Neural-Probabilistic Model for Authorship Verification Under Covariate Shift. *arXiv preprint arXiv:2106.11196*. [↑](#endnote-ref-3)
4. Tran, H. N., & Kruschwitz, U. (2021). ur-iw-hnt at GermEval 2021: An Ensembling Strategy with Multiple BERT Models. *arXiv preprint arXiv:2110.02042*. [↑](#endnote-ref-4)
5. Böck, J., Liakhovets, D., Schütz, M., Kirchknopf, A., Slijepčević, D., Zeppelzauer, M., & Schindler, A. (2021, September). AIT\_FHSTP at GermEval 2021: Automatic Fact Claiming Detection with Multilingual Transformer Models. In *Proceedings of the GermEval 2021 Shared Task on the Identification of Toxic, Engaging, and Fact-Claiming Comments* (pp. 76-82). [↑](#endnote-ref-5)
6. Gémes, K., & Recski, G. (2021, September). TUW-Inf at GermEval2021: Rule-based and Hybrid Methods for Detecting Toxic, Engaging, and Fact-Claiming Comments. In Proceedings of the GermEval 2021 Shared Task on the Identification of Toxic, Engaging, and Fact-Claiming Comments (pp. 69-75). [↑](#endnote-ref-6)
7. Schmidhuber, M. (2021, September). Universität Regensburg MaxS at GermEval 2021 Task 1: Synthetic Data in Toxic Comment Classification. In *Proceedings of the GermEval 2021 Shared Task on the Identification of Toxic, Engaging, and Fact-Claiming Comments* (pp. 62-68). [↑](#endnote-ref-7)
8. Schütz, M., Demus, C., Pitz, J., Probol, N., Siegel, M., & Labudde, D. (2021, September). DeTox at GermEval 2021: Toxic Comment Classification. In *Proceedings of the GermEval 2021 Shared Task on the Identification of Toxic, Engaging, and Fact-Claiming Comments* (pp. 54-61). [↑](#endnote-ref-8)
9. Haak, F., & Engelmann, B. (2021, September). IRCologne at GermEval 2021: Toxicity Classification. In *Proceedings of the GermEval 2021 Shared Task on the Identification of Toxic, Engaging, and Fact-Claiming Comments* (pp. 47-53). [↑](#endnote-ref-9)
10. Ratner, A., Bach, S. H., Ehrenberg, H., Fries, J., Wu, S., & Ré, C. (2017, November). Snorkel: Rapid training data creation with weak supervision. In *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases* (Vol. 11, No. 3, p. 269). NIH Public Access. [↑](#endnote-ref-10)
11. Panda, S., & Levitan, S. I. (2021, September). HunterSpeechLab at GermEval 2021: Does Your Comment Claim A Fact? Contextualized Embeddings for German Fact-Claiming Comment Classification. In *Proceedings of the GermEval 2021 Shared Task on the Identification of Toxic, Engaging, and Fact-Claiming Comments* (pp. 100-104). [↑](#endnote-ref-11)
12. Akomeah, K. O., Kruschwitz, U., & Ludwig, B. (2021, September). UR@ NLP\_A\_Team@ GermEval 2021: Ensemble-based Classification of Toxic, Engaging and Fact-Claiming Comments. In *Proceedings of the GermEval 2021 Shared Task on the Identification of Toxic, Engaging, and Fact-Claiming Comments* (pp. 95-99). [↑](#endnote-ref-12)
13. Kruschwitz, U., & Hoffmann, J. (2020). UR\_NLP@ HaSpeeDe 2 at EVALITA 2020: Towards Robust Hate Speech Detection with Contextual Embeddings. [↑](#endnote-ref-13)
14. Arjun, T. H., Arvindh, A., & Ponnurangam, K. (2021, September). Precog-LTRC-IIITH at GermEval 2021: Ensembling Pre-Trained Language Models with Feature Engineering. In *Proceedings of the GermEval 2021 Shared Task on the Identification of Toxic, Engaging, and Fact-Claiming Comments* (pp. 39-46). [↑](#endnote-ref-14)
15. Gawron, C., & Schmidt, S. (2021). FH-SWF SG at GermEval 2021: Using Transformer-Based Language Models to Identify Toxic, Engaging, & Fact-Claiming Comments. *arXiv preprint arXiv:2109.02966*. [↑](#endnote-ref-15)
16. Morgan, Skye, Tharindu Ranasinghe, and Marcos Zampieri. "WLV-RIT at GermEval 2021: Multitask Learning with Transformers to Detect Toxic, Engaging, and Fact-Claiming Comments." *arXiv preprint arXiv:2108.00057* (2021). [↑](#endnote-ref-16)
17. Chan, B., Schweter, S., & Möller, T. (2020). German's Next Language Model. *arXiv preprint arXiv:2010.10906*. [↑](#endnote-ref-17)
18. Bornheim, T., Grieger, N., & Bialonski, S. (2021). FHAC at GermEval 2021: Identifying German toxic, engaging, and fact-claiming comments with ensemble learning. *arXiv preprint arXiv:2109.03094*. [↑](#endnote-ref-18)
19. Risch, J., & Krestel, R. (2018, August). Aggression identification using deep learning and data augmentation. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)* (pp. 150-158). [↑](#endnote-ref-19)
20. Risch, J., & Krestel, R. (2020, May). Bagging BERT models for robust aggression identification. In *Proceedings of the Second Workshop on Trolling, Aggression and Cyberbullying* (pp. 55-61). [↑](#endnote-ref-20)
21. Risch, J., & Krestel, R. (2020, May). Top comment or flop comment? predicting and explaining user engagement in online news discussions. In *Proceedings of the International AAAI Conference on Web and Social Media* (Vol. 14, pp. 579-589). [↑](#endnote-ref-21)
22. Malmasi, S., & Zampieri, M. (2018). Challenges in discriminating profanity from hate speech. *Journal of Experimental & Theoretical Artificial Intelligence*, *30*(2), 187-202. [↑](#endnote-ref-22)
23. Zampieri, M., Malmasi, S., Nakov, P., Rosenthal, S., Farra, N., & Kumar, R. (2019). Predicting the type and target of offensive posts in social media. *arXiv preprint arXiv:1902.09666*. [↑](#endnote-ref-23)
24. Ranasinghe, T., Zampieri, M., & Hettiarachchi, H. (2019, December). BRUMS at HASOC 2019: Deep Learning Models for Multilingual Hate Speech and Offensive Language Identification. In *FIRE (Working Notes)* (pp. 199-207). [↑](#endnote-ref-24)
25. Zhao, Z., Zhang, Z., & Hopfgartner, F. (2021, April). A Comparative Study of Using Pre-trained Language Models for Toxic Comment Classification. In *Companion Proceedings of the Web Conference 2021* (pp. 500-507). [↑](#endnote-ref-25)
26. d'Sa, A. G., Illina, I., & Fohr, D. (2020, February). Bert and fasttext embeddings for automatic detection of toxic speech. In *2020 International Multi-Conference on:“Organization of Knowledge and Advanced Technologies”(OCTA)* (pp. 1-5). IEEE. [↑](#endnote-ref-26)
27. Maslej-Krešňáková, V., Sarnovský, M., Butka, P., & Machová, K. (2020). Comparison of deep learning models and various text pre-processing techniques for the toxic comments classification. *Applied Sciences*, *10*(23), 8631. [↑](#endnote-ref-27)
28. Safaya, A., Abdullatif, M., & Yuret, D. (2020, December). Kuisail at semeval-2020 task 12: Bert-cnn for offensive speech identification in social media. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation* (pp. 2054-2059). [↑](#endnote-ref-28)
29. Song, G., Huang, D., & Xiao, Z. (2021). A Study of Multilingual Toxic Text Detection Approaches under Imbalanced Sample Distribution. *Information*, *12*(5), 205. [↑](#endnote-ref-29)
30. Hu, J., Ruder, S., Siddhant, A., Neubig, G., Firat, O., & Johnson, M. (2020, November). Xtreme: A massively multilingual multi-task benchmark for evaluating cross-lingual generalisation. In *International Conference on Machine Learning* (pp. 4411-4421). PMLR. [↑](#endnote-ref-30)
31. Weller-Di Marco, Marion. "Simple compound splitting for German." *Proceedings of the 13th Workshop on Multiword Expressions (MWE 2017)*. 2017. [↑](#endnote-ref-31)
32. Koehn, Philipp, and Kevin Knight. "Empirical methods for compound splitting." *arXiv preprint cs/0302032* (2003). [↑](#endnote-ref-32)
33. Bay, Matthias, et al. "Term Extraction from Medical Documents Using Word Embeddings." *2020 6th IEEE Congress on Information Science and Technology (CiSt)*. IEEE, 2021. [↑](#endnote-ref-33)
34. Krotova, Irina, Sergey Aksenov, and Ekaterina Artemova. "A Joint Approach to Compound Splitting and Idiomatic Compound Detection." *arXiv preprint arXiv:2003.09606* (2020). [↑](#endnote-ref-34)
35. Ma, Jianqiang, Verena Henrich, and Erhard Hinrichs. "Letter sequence labeling for compound splitting." *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*. 2016. [↑](#endnote-ref-35)
36. Chauhan, D. S., Dhanush, S. R., Ekbal, A., & Bhattacharyya, P. (2020, July). Sentiment and emotion help sarcasm? a multi-task learning framework for multi-modal sarcasm, sentiment and emotion analysis. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (pp. 4351-4360). [↑](#endnote-ref-36)
37. Barnes, J., Velldal, E., & Øvrelid, L. (2021). Improving sentiment analysis with multi-task learning of negation. *Natural Language Engineering*, *27*(2), 249-269. [↑](#endnote-ref-37)
38. Malmasi, S., & Zampieri, M. (2018). Challenges in discriminating profanity from hate speech. *Journal of Experimental & Theoretical Artificial Intelligence*, *30*(2), 187-202. [↑](#endnote-ref-38)
39. <https://sites.google.com/view/germeval2017-absa/> [↑](#endnote-ref-39)
40. Feng, Steven Y., et al. "A survey of data augmentation approaches for nlp." *arXiv preprint arXiv:2105.03075* (2021). [↑](#endnote-ref-40)
41. Boyd, Adriane. "Using Wikipedia edits in low resource grammatical error correction." *Proceedings of the 2018 EMNLP Workshop W-NUT: The 4th Workshop on Noisy User-generated Text*. 2018. [↑](#endnote-ref-41)
42. Şahin, Gözde Gül, and Mark Steedman. "Data augmentation via dependency tree morphing for low-resource languages." arXiv preprint arXiv:1903.09460 (2019). [↑](#endnote-ref-42)
43. Yun, Sangdoo, et al. "Cutmix: Regularization strategy to train strong classifiers with localizable features." Proceedings of the IEEE/CVF International Conference on Computer Vision. 2019. [↑](#endnote-ref-43)
44. Ghiasi, Golnaz, et al. "Simple copy-paste is a strong data augmentation method for instance segmentation." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021. [↑](#endnote-ref-44)
45. Ng, Nathan, Kyunghyun Cho, and Marzyeh Ghassemi. "Ssmba: Self-supervised manifold based data augmentation for improving out-of-domain robustness." arXiv preprint arXiv:2009.10195 (2020). [↑](#endnote-ref-45)
46. Howard, Jeremy, and Sebastian Ruder. "Universal language model fine-tuning for text classification." *arXiv preprint arXiv:1801.06146* (2018). [↑](#endnote-ref-46)
47. Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems*. 2017. [↑](#endnote-ref-47)
48. https://github.com/deepset-ai/FARM [↑](#endnote-ref-48)