Latent Feature-Based Data Splits to Improve Generalisation Evaluation

A Hate Speech Detection Case Study

Maike Züfle

Master of Science

School of Informatics

University of Edinburgh

2023

Abstract

Machine learning models are expected to work well on new data they have not seen during training. This ability is often called generalisation. To test this in advance, a model is usually evaluated on unseen test data split randomly from the dataset. However, random splits have been proven to overestimate the performance of models. This work introduces a more challenging data split to improve generalisation evaluation, specifically addressing out-of-distribution generalisation in Natural Language Processing. The method leverages the internal representations of language models, capturing task-specific properties of the data that are most important for the model. Hate speech detection, a socially-relevant task, where generalisation is critical, is used as an illustrative task to develop and test the proposed splitting method. The proposed split reveals dramatic drops in test set performance in comparison to a random split for different models and datasets. Moreover, while the data split is always based on a specific model, it proves similarly challenging for other models. Therefore, creating a data split in the proposed way is an effective way to evaluate the robustness of language models.

Research Ethics Approval

This project obtained approval from the Informatics Research Ethics committee.
Ethics application number: 758105
Date when approval was obtained: 2023-05-17

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Maike Züfle*)

Acknowledgements

I would like to thank my supervisors Prof. Ivan Titov and Verna Dankers for their kind support, constructive advice and invaluable guidance throughout this project.

Table of Contents

Chapter 1

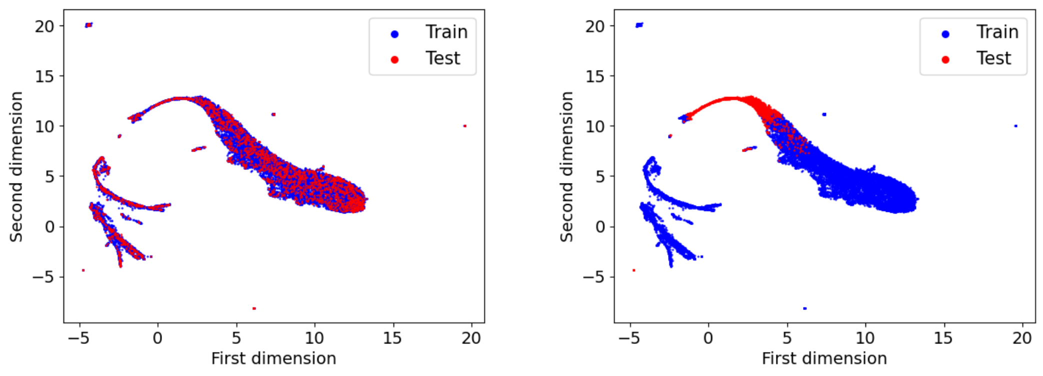
Introduction

Machine learning models are usually deployed on data the model has not seen before. Therefore, models are trained not only to perform well on the data they encounter during training but also on new, unseen data. This ability to “successfully transfer representations, knowledge, and strategies” is referred to as generalisation (Hupkes et al., 2023, p.2).

In Natural Language Processing (NLP), unseen data can include text from different time periods (Lazaridou et al., 2021), authors (Huang and Paul, 2019), and dialects (Ziems et al., 2022) as well as words and phrases that were not seen during training (Elangovan, He, and Verspoor, 2021). The data that a model encounters after training is often unknown a priori. Performing successfully on this data in spite of such distributional changes is called out-of-distribution (o.o.d.) generalisation. In this work, this will mostly be referred to simply as generalisation. With all these potential data shifts, how can the ability to generalise be measured? Despite a lack of consensus in the field (Hupkes et al., 2023), common ground is that generalisation should be evaluated on a test set and that this test set needs to be different from the training data.

This is frequently achieved by splitting a dataset into train, validation and test sets. The training data is used to optimise the parameters, often called weights, of the model. The validation data is used to select the hyperparameters that control the training. Finally, the test data is used to measure the generalisation performance of the model.

Researchers often propose a standardised data split when introducing new datasets. This enables a straightforward and comparable way to measure models’ abilities to generalise. However, Gorman and Bedrick (2019) argue that standardised splits overestimate a model’s performance in comparison to splitting the data randomly.



(a) Random Split (b) New Split

Figure 1.1: Comparison between a random split and the proposed (new) data split based on hidden representations of a language model. The plots show the hidden representation of all examples in the dataset reduced to two dimensions by UMAP (McInnes, Healy, and Melville, 2020).

A random split assumes that the data are independent and identically distributed (i.i.d.) and that the split therefore suffices to evaluate models' generalisation capabilities. Unfortunately, this approach will likely also overestimate model performance in practice (Søgaard et al., 2021), as the i.i.d. assumption often does not hold when dealing with distributions that shift over time (Lazaridou et al., 2021) or come from different applications (Malinin et al., 2021). The assumption also does not hold when there are sampling dependencies, for example, several posts from the same author or from the same conversation (Arango, Pérez, and Poblete, 2019).

This work aims to go beyond the random split method by developing a process that produces more challenging data splits, which can better evaluate generalisation performance. A particularly hard split will approximate the worst distributional shifts that might arise in practice. In this way, it can help with measuring just how well models will perform in worst cases, and therefore also with developing models that are more robust to distributional shifts. However, often it is hard to detect what data is out-of-distribution (Arora, Huang, and He, 2021), and especially what distributional shifts might be the most challenging for a specific model.

To humans, properties of the data such as input length (Varis and Bojar, 2021), spelling mistakes (Ebrahimi et al., 2018) or overlap between excerpts of sentences (Broscheit, Do, and Gaspers, 2022) might seem challenging and have indeed been examined. However, other properties of the data might only be challenging for language models and not necessarily for humans. Therefore, other work has relied on the outputs of a model, such as the log-likelihood of predictions (Godbole and Jia, 2022), to produce challenging splits. This work, in contrast, lies between relying on the surface form of the inputs and relying on the outputs of a model. A model's internal representations are leveraged to create a data split specifically designed for the provided model. Internal representations capture what is important to a model, so separating examples in the model's internal embedding space is likely to create more challenging splits than relying directly on the inputs or outputs of a model. In computer vision, latent representations are used to detect o.o.d. examples in a dataset (Rabanser, Günnemann, and Lipton, 2019; Roth et al., 2022), making a split based on these likely to be challenging.

To obtain the train-test split, the internal representations are clustered and whole clusters are assigned to either the train or the test set (whilst maintaining the overall target class distributions). Fig. 1.1 provides an illustrative example of how a data split like that can differ from an i.i.d. split. Clustering has been used to split datasets in prior work by Broscheit, Do, and Gaspers (2022). However, the authors there cluster n-gram feature representations, rather than hidden representations.

Hate speech detection is used as the testing ground for developing the splitting approach. The term *hate speech* describes a "deliberate attack directed towards a specific group of people motivated by aspects of the group's identity" (de Gibert et al., 2018, p. 13). Detection of hate speech is not only socially relevant but also interesting for the problem at hand because performance on this task is often overestimated. Since the advent of large language models, performance on hate speech detection datasets gives the impression that the problem is close to being solved (Vidgen et al., 2019). However, despite this progress, recent work points out that hate speech detection models still suffer from generalisation problems, such as overfitting to specific terms (Dixon et al., 2018; Waseem, Thorne, and Bingel, 2018; Kennedy et al., 2020; Palmer et al., 2020), or specific social media users (Arango, Pérez, and Poblete, 2019). This work aims to develop a data-splitting method that can reveal this discrepancy. The proposed method is developed on a binary classification task based on hate speech data from Reddit (Qian et al., 2019) and later tested on a three-way classification task based on hate speech data from Gab and Twitter (Mathew et al., 2020).

Training state-of-the-art models to detect hate speech with the proposed data split reveals significantly decreased generalisation performance on the test set in comparison to using a random split. The resulting data split leads to a harder test set, while the train set is not *incomplete* in the sense that models trained on this train set still perform well on independent test data. Moreover, the data splits that are obtained with the hidden representations of a particular model are also challenging for other language models, demonstrating the potential for using a single split to compare different models. Apart from providing a more accurate assessment of the generalisation capabilities of existing models, the splits could also be helpful in the development of more robust model architectures and training techniques in future work, thereby contributing to the large body of research in generalisation improvement (Feng et al., 2021; Pyysalo et al., 2021; Naik, Lehman, and Rosé, 2022, among others).

Note that many works in this area focus on the more specific problem of domain adaptation (Miller, Laparra, and Bethard, 2021; Ramesh Kashyap et al., 2021; Nishida and Matsumoto, 2022, among others). While the proposed method could split a dataset along domain lines, this is neither the goal of the approach nor is it explicitly encouraged. Additionally, domain adaptation often knows the target domain a priori, and work focuses on adapting a trained model to the new domain. In contrast, the proposed data-splitting approach is designed to evaluate the general robustness of models to unforeseen distributional shifts.

The rest of this work is structured as follows. Chapter 2 takes a deeper look at prior work on generalisation and hate speech detection. Chapter 3 introduces the proposed data-splitting approach in detail, including the clustering methods used, the datasets, and the experimental setup. Chapter 4 presents model evaluation results on the produced data splits, Chapter 5 analyses the splits in detail, and Chapter 6 finishes with a discussion and conclusion.

Warning: This work includes posts that contain hate speech, which may be offensive or upsetting to some readers.

Chapter 2

Background and Related Work

2.1 Generalisation

A lot of work points in the direction that models with high or even human-like scores (Chowdhery et al., 2022, among others) on i.i.d. splits do not generalise as robustly as the results would suggest (Lake and Baroni, 2017; McCoy, Pavlick, and Linzen, 2019; Kim and Linzen, 2020; Sinha et al., 2021; Razeghi et al., 2022). For example, McCoy, Pavlick, and Linzen (2019) demonstrate that models overly rely on heuristics, such as the appearance of certain words, to make their predictions. Gardner et al. (2020) also find that models sometimes depend on simple decision rules to perform well on test data, without actually understanding the task. This is further supported by Kaushik, Hovy, and Lipton (2019), who find that classifiers with impressive results in sentiment analysis and natural language inference (NLI) tasks often fail on examples from the dataset that are modified to carry an opposite meaning. Perhaps less surprising is that models tend not to generalise well to different domains (Michel and Neubig, 2018; Malinin et al., 2021).

An alternative to standardised or random splits is to gather new data to test generalisation. This can be data from a different domain; for example, for the popular NLP benchmarks GLUE (Wang et al., 2018) and Super-GLUE (Wang et al., 2019) an out-of-distribution test set has been presented by Yang et al. (2022). Another possibility to gather new data is to synthetically create a test set that is linguistically inspired by using formal languages and different levels of recursions in the test set (Kim and Linzen, 2020). However, this does not represent *natural* language generalisation scenarios.

Another popular way to gather new data for testing generalisation is to create adversarial examples, i.e., examples that are designed to trick the model. Human-written adversarial examples for various NLP tasks, including hate speech detection, are collected in the Dynabench platform (Kiela et al., 2021). However, there are also automatically generated adversarial examples (Zhang et al., 2020). Task-specific adversarial datasets include datasets for NLI with special kinds of negation (Gururangan et al., 2018) or based on heuristics (McCoy, Pavlick, and Linzen, 2019). For text classification, adversarial examples can be made by changing only one character of the input (Chen et al., 2019) or by swapping characters and tokens in the input based on a model's gradients (Ebrahimi et al., 2018). All of these approaches lead to lower scores and reveal different flaws in language models.

Creating new data in the above ways is a fruitful way to test generalisation, but data collection and annotation are costly. Splitting existing datasets in a non-i.i.d. manner is a more efficient approach. Such a non-i.i.d. data split can be based on the input of the language models: For example, a data split that minimises the word overlap between the train and test sets already presents a challenge for language models (Elangovan, He, and Verspoor, 2021). Familiar words appearing in new contexts in the test set can also be challenging (Keysers et al., 2019). Similar findings hold for data splits based on dependency parsing, where new linguistic structures appear in the test set (Søgaard, 2020). Along the same lines, Broscheit, Do, and Gaspers (2022) split a spoken language understanding dataset based on the distributional drift in the input for a language model. Yet another way to re-split the dataset is to test temporal generalisation by using documents with a more recent timestamp in the test set (Lazaridou et al., 2021).

Generalisation can also be tested with output-based non-i.i.d. splits: In NLI, Naik et al. (2018) analyse the predictions of a model to find challenging phenomena in the dataset and they then use these phenomena to construct test sets. For text classification tasks, Godbole and Jia (2022) re-split a dataset based on the predicted log-likelihood for each example, using low-likelihood examples in the test set.

The data split method proposed in this work lies in between the input-based and output-based data-splitting methods, relying on the internal representations of the model.

2.2 Hate Speech

Hate speech detection is used as the testbed for developing the proposed generalisation evaluation approach. *Hate speech* is widely used as an umbrella term for harmful, abusive and offensive language (Waseem et al., 2017). The authors divide hate speech along two dimensions: explicit-implicit and directed-generalised. For example, *"Go kill yourself"* is explicit and directed, whereas *"most of them come north and are good at just mowing lawns"* is implicit and generalised (Waseem et al., 2017, p.3). Another subdivision of hate speech, specifically of harmful stereotypes, is given by the *Stereotype Content Model* (Fiske et al., 2002, SCM). SCM classifies stereotypes along two dimensions: the intent of the target group (friendly or hostile), and the ability of the target group to carry out its intent. Fraser, Nejadgholi, and Kiritchenko (2021) develop a model to map hate speech to SCM classes. However, not all hate speech involves stereotypes; it also includes objectification and threats (Anzovino, Fersini, and Rosso, 2018).

There exists a wide range of hate speech datasetss: Hate speech can come from different domains and address various topics, such as *gender, religion, nationality,* or *race*. Some datasets are *topic-specific*, whilst others are *topic-generic*. Topic-generic datasets encompass a wider range of topics and since work on hate speech detection has been growing (Vidgen et al., 2019), multiple topic-generic benchmark datasets (Waseem, 2016; Founta et al., 2018; Vidgen et al., 2021a, and others) have popped up. Most of these datasets rely on data from social media platforms, such as Reddit (Qian et al., 2019; Vidgen et al., 2021a), Twitter (ElSherief et al., 2021), Gab (Qian et al., 2019; Mathew et al., 2020), or Stormfront (de Gibert et al., 2018), but synthetic datasets also exist (Vidgen et al., 2021b). On the other hand, topic-specific datasets focus on a particular topic or selection of topics, such as hate speech against women (Fersini, Nozza, and Rosso, 2018), or hate speech against immigrants and women (Basile et al., 2019). This work uses topic-generic datasets, as these offer a wider distribution of examples and generalisation across them is more interesting to look at.

The hate speech datasets are designed for different purposes: Hate speech tasks can involve detection, identification of the topic or target group, or generation of responses (Qian et al., 2019). This work is restricted to hate speech classification; an example in the dataset can be assigned a positive (*hate*) label or a negative (*noHate*) label.

Hate speech detection can be a challenging task, especially when using data from social media platforms. Examples often include spelling mistakes, colloquial terms, humour, irony, sarcasm (Vidgen et al., 2019), and hate symbols (Qian et al., 2019). However, recent advances in NLP, including contextualised word embeddings and large language models, have led to impressive results (Fortuna and Nunes, 2018; Vidgen et al., 2019). State-of-the-art models mostly involve the use of transformers (Vaswani et al., 2017; Liu, Li, and Zou, 2019). For example, Caselli et al. (2021) further pretrain a BERT model (Devlin et al., 2019) using hate speech data prior to fine-tuning on hate speech classification tasks.

Despite impressive results, non-i.i.d. generalisation has proven to be challenging for hate speech detection (Yin and Zubiaga, 2021). One problem when generalising is the low proportion of hate examples in a dataset; models trained on a more balanced class distribution often generalise better even if the distribution does not mirror reality (Swamy, Jamatia, and Gambäck, 2019). Hate speech models also seem to overfit easily. Arango, Pérez, and Poblete (2019) find that models tend to overfit to specific users and perform well on the test set simply because it contains posts from the same users as the train set. Other traps for overfitting are the use of specific terms (Dixon et al., 2018; Kennedy et al., 2020) or slurs and pejorative terms (Waseem, Thorne, and Bingel, 2018; Kurrek, Saleem, and Ruths, 2020; Palmer et al., 2020) in hate speech. Therefore, when hate speech does not involve overt swear words or triggering words, then models often fail to detect hate speech (ElSherief et al., 2021). Models also do not generalise well between different sources of data, such as when switching between examples from Fox News, Twitter or Wikipedia (Karan and Šnajder, 2018). Talat, Thorne, and Bingel (2018) point out that differing data sampling and annotation methodologies can make generalisation evaluation difficult, especially when annotators and researchers come from different cultures and have different understandings of hate speech. Nejadgholi and Kiritchenko (2020) find topic biases in topic-generic datasets also lead to challenging generalisation scenarios, and Bourgeade et al. (2023) examine topic-generic and topic-specific generalisation in hate speech, testing models trained on topic-generic and topic-specific datasets on different topics of hate speech.

In response to these generalisation issues, recent works combine existing hate speech datasets in order to improve generalisation performance (Fortuna, Bonavita, and Nunes, 2018; Salminen et al., 2020; Chiril et al., 2022; Bourgeade et al., 2023). However, this inevitably leads to larger datasets, in turn leading to higher computational effort. Moreover, Nejadgholi and Kiritchenko (2020) find that the inconsistent definition of hate-speech across datasets makes generalisation hard to evaluate, so combining existing datasets is a challenging task in itself.

While many of these approaches are effective ways to evaluate generalisation, they are also costly. Augmenting datasets requires more data, and evaluating whether a model overfits to different users, posts, data sources etc. requires annotations of these properties. However, these characteristics are often not available due to privacy requirements or because the annotations were simply not included with the dataset release. Therefore, this work aims to find a data split that can evaluate generalisation without any such annotations, relying instead only on a model's internal representations.

Chapter 3

Dataset and Methodology

The goal of this work is to contribute to a better evaluation of o.o.d. generalisation. Therefore, this work proposes data-splitting methods that lead to challenging data splits. The datasets used are introduced in Section 3.1 and the methodology to obtain the splits is presented in Section 3.2. The datasets concern hate speech detection.

3.1 Datasets

This section presents two hate speech datasets. The first is used as a *development dataset* to develop the splitting method(s), find working hyperparameter settings and choose interesting data splits. To ensure that the chosen splitting method does not overfit on this particular dataset, it is then evaluated on another dataset, the *test dataset*.

3.1.1 Development Dataset

The proposed data-splitting method is developed on the Reddit hate speech detection dataset from Qian et al. (2019). This dataset is topic-generic and as such contains a wider range of examples than a topic-specific dataset. This makes it a good testbed for analysing the robustness and generalisation ability of language models. Moreover, with a size of 22,317 examples, it is large enough to draw meaningful conclusions without leading to prohibitively long training times. Each example in the dataset is labelled as either *hate* (23.5%) or *noHate* (76.5%). The dataset was collected from ten different subreddits by retrieving potential hate speech posts using hate keywords taken from ElSherief et al. (2018). The hate keywords correspond roughly to the following categories: *archaic*, *class*, *disability*, *ethnicity*, *gender*, *nationality*, *religion*, and *sexual* *orientation*. The data is structured in conversations that consist of at most 20 comments by the same or different authors. These comments were manually annotated with *hate* or *noHate*, with each annotator assigned five conversations.

3.1.2 Test Dataset

The HateXplain dataset (Mathew et al., 2020) is used to test the data-splitting method. It contains 20,148 examples taken from Twitter and Gab. Posts from the combined collection were filtered based on a lexicon of hate keywords and phrases by Davidson et al. (2017), Mathew et al. (2019), and Ousidhoum et al. (2019). The selected posts were then manually annotated. In contrast to the development dataset, HateXplain examples are labelled as either *hateful* (31%), *offensive* (29%) or *normal* (40%), as proposed by Davidson et al. (2017). Offensive speech differs from hate speech in that it uses offensive terms, but without directing these against any person or group in particular. The following examples from the dataset illustrate this difference: *fuck the nigger bitches* (hate), *lmao gay haircut* (offensive).

All offensive and hate examples are annotated with the group or community that they affect or target. These communities include, among others, *Africans*, *Jewish People*, *Homosexuals* and *Women*. For example, the hate post above is annotated with the communities *African, Woman*, whereas the offensive post is annotated with *Homosexuals*. While the models are still trained on the three-way classification task, the communities are used for further analysis in Chapter 5. The possibility to use the communities for this analysis is one of the reasons for choosing this dataset as the test dataset, alongside being topic-generic and of a feasible size.

3.2 Methodology

This section describes the proposed data-splitting approach for creating challenging train-test splits in order to improve the evaluation of o.o.d. generalisation. The pipeline can be described in three steps. In Step 1, the method extracts the latent representations of inputs from a language model that was finetuned on the task. In Step 2, the data is clustered based on these representations and clusters are assigned to either the train or the test set. In Step 3, language models are then trained and evaluated on this new train-test-split. Additionally to the obtained test set, the language models are also evaluated on independent test data, that was set aside for this purpose. An overview

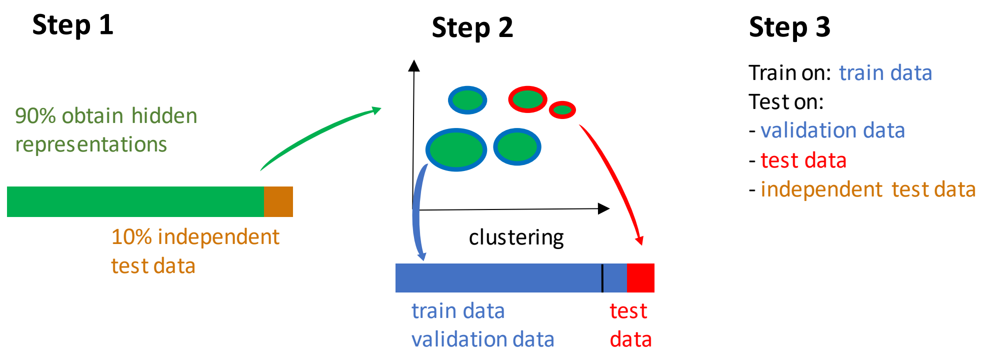


Figure 3.1: Pipeline of the proposed data-splitting approach.

of this pipeline is illustrated in Fig. 3.1. The key idea behind the approach is to create train and test sets with different semantic properties relevant to the task. Since language models store semantic properties in their hidden representations, inputs with similar properties are placed closer in the embedding space (Thompson and Mimno, 2020; Grootendorst, 2022). Therefore, assigning clusters to the train and test sets separates the examples in the embedding space.

In what follows, the different steps are explained in detail. Section 3.2.1 describes different ways of obtaining latent representations (Step 1), and Section 3.2.2 outlines different clustering and splitting approaches (Step 2). Section 3.2.3 introduces the language models used to obtain the representations and to evaluate the splitting method (Step 3). Finally, Section 3.3 gives an overview of the experimental setup.

3.2.1 Obtaining Latent Representations

The internal representations of language models are used to cluster and split the data. To obtain task-specific representations, a language model is first finetuned for the given task. Note that some of the data (10%) are needed as a validation set for finetuning in order to optimise hyperparameters and decide when to stop training. This validation set is not used any further for creating the data split. The train-test split is based on the remaining 90% of the data. However, the finetuning validation data is used later for further analysis as an independent data sample (Section 4.2.3). To avoid any confusion, the finetuning validation data is called *independent test data*, depicted in orange in the pipeline overview in Fig. 3.1.

The finetuned language model is then used to extract latent representations. But how can a single latent representation be obtained for each input example? This work uses transformer language models (Vaswani et al., 2017; Devlin et al., 2019; Liu et al., 2019)

(Section 3.2.3 elaborates on the exact models used). Transformers map each (sub)word of the input to context-sensitive representations. Each layer of the model maintains the input length and outputs a corresponding representation for each (sub)word of the input. The input to the model consists of the tokenised input sentence, along with a special symbol, the *class token*. The latent representation of the class token after the final layer is then used to decide on the class for the entire input. The representation of the class token is also commonly used as a representation for the whole input sentence (May et al., 2019; Qiao et al., 2019, among others), and this work does the same.

There exist other ways to extract sentence embeddings from Transformer models such as Sentence-BERT (Reimers and Gurevych, 2019). However, since this work is interested in the task-specific latent representations of the model and not in the sentence embedding itself, the latent representation of the class token is used.

Latent representations from language models are usually high-dimensional. However, when measuring the distance of points in high-dimensional data, distance metrics tend to fail to capture the concept of proximity or distance (Beyer et al., 1999; Aggarwal, Hinneburg, and Keim, 2001). Moreover, distance metrics applied to full-dimensional transformer-based hidden representations can be overly influenced by individual dimensions (Timkey and Schijndel, 2021). Since distance measurement is important for clustering, experiments are also conducted with lower dimensional representations. Specifically, the latent representations for the data split are obtained in the following three ways:

1. The latent representations from the last hidden layer before the classification layer of the language model are extracted. This yields a representation for each input word, including the additional class token. Only the class token representation is kept.
2. The latent representations are obtained as in 1. and are then embedded into lower-dimensional spaces. The non-linear dimensionality reduction algorithm UMAP (McInnes, Healy, and Melville, 2020) is used as it presents no computational restrictions on the dimension of the embedding space and McInnes, Healy, and Melville (2020) argue that it preserves the global structure of the data better than other popular dimensionality reduction algorithms. UMAP has also been used in other recent NLP literature for this purpose (Grootendorst, 2022). The UMAP algorithm can be roughly described by two steps: A graph representation of the data is computed and then a lower dimensional representation is obtained by finding a close topological representation to the original one, optimised with a cross-entropy loss.

3. After the last hidden layer, and before the classification layer, an additional *bottleneck* is added to the model. The bottleneck consists of a linear layer that compresses the latent representations into a smaller dimension. As in 1., only the representation for the class token is extracted. This dimensionality reduction approach is more in line with the goal of this work, i.e., using the hidden representations of the model as task-specific features. In contrast, UMAP is task-agnostic.

In the first case, where the full representations are extracted, the latent representations have $d = 768$ dimensions for a standard BERT model (Devlin et al., 2019). For the second and third cases, this work experiments with three different dimensions, $d = 10$, $d = 50$ and $d = 200$. The following naming convention is used for splits: [model]-[dimensionality reduction method]-[dimension]. For example, BERT-bottleneck-50 refers to splitting based on the 50-dimensional bottleneck representations of a BERT model. Splits without dimensionality reduction are referred to as [model]-full-dimensional.

3.2.2 Clustering and Splitting the Data

The previous section presented Step 1 of the pipeline – obtaining latent representations. This section presents Step 2 – clustering the data based on the latent representations and assigning the clusters to the train and test sets.

Each representation from Step 1 gives the position of an input example in the latent embedding space. The examples are clustered in this space using the k-means algorithm (Lloyd, 1982). K-means is chosen due to its efficiency and simplicity, and its use in previous work on clustering language model embeddings (Sia, Dalmia, and Mielke, 2020). Since k-means is not a deterministic algorithm (the initial cluster centres are chosen randomly), results are reported for three different random seeds. Hyperparameters of the k-means clustering can be found in Table A.3

After clustering, each cluster is assigned to either the train or the test set. In order to obtain a data split that is suitable for evaluating generalisation on the given task, two constraints are important when assigning the clusters: 1) The test data has a fixed size: This work chooses the size of the test set to be 10% of the data. 2) The train and test set need to have equal class distributions: Class imbalance usually leads to lower performance for the minority class and using equal class distributions in the train and test sets excludes class imbalance as a confounding factor when examining the performance of the novel data splits. A partition of the dataset that fulfils these constraints will be referred to as *target* in this work.

Reaching a target test set is not straightforward when assigning clusters to the train and test sets: Clusters might be too big for the test set or contain only examples of one class. Therefore, two different algorithms are designed to decide how to split the clusters to reach a target test set. The first approach aims to assign whole clusters to the train and test sets, dividing up as few clusters as possible. This approach will be referred to as *subset-sum-split*. The second approach, referred to as *closest-split*, aims to put as much distance as possible between the train and test set examples. This is done by placing clusters that are spatially close into the same set. Note that both algorithms lead to an under-representation of parts of the latent space in the model's training set, but while the subset-sum-split achieves this locally, the closest-split achieves this globally. The algorithms are explained in detail below.

3.2.2.1 Algorithm 1: Subset-Sum-Split

The aim of the subset-sum-split is to assign whole clusters to the train and test sets and, if possible, not to divide a cluster between the two. When a cluster is divided between the train and test sets, the model can learn how to deal with the cluster more easily.

The constraints on the class and test ratios explained above, and the additional constraint of keeping whole clusters together can be described by the Subset Sum Problem (Kellerer, Pferschy, and Pisinger, 2004). The standard Subset Sum Problem is to decide whether a subset of a list of natural numbers exists that sums up to a given target value. This problem is NP-hard but can be solved with dynamic programming in pseudo-polynomial time. In this setting, the Subset Sum Problem can be modified to a multidimensional Subset Sum Problem: The multidimensional target consists of the number of desired test examples for each class in the dataset, for example, the target $(25, 75)$ means that the test set should contain 25 hate and 75 noHate examples. The task is then to select a subset of the clusters, such that the number of examples for each class sums up to the desired target. To improve the chances of reaching the desired target, the Subset Sum Problem is solved for $k = 3$ to $k = 50$ clusters and the solution closest to the desired target using the smallest k is taken as the test set. If the closest solution does not match the exact target sum, examples from another cluster are used to complete the test set. The algorithm is summarised in Algorithm 1. Note that the clusters in the test set do not necessarily lie close to each other in space, as this is not a constraint for this algorithm .

Algorithm 1 Subset-Sum-Split

```
1: Input: $n$ latent representations Set $X = \{x_i\}_{i=1}^{n}$
2: target_sum ← Tuple ($n \cdot$ hate ratio $\cdot$ test ratio, $n \cdot$ noHate ratio $\cdot$ test ratio)
3: split_shortlist ← Set
4: for $c = 2$ to Maximum Number of Clusters do
5:     clusters ← List kmeans($X$, $c$)
6:     class_frequencies ← List get_class_frequencies(clusters)    ▷ List of tuples
7:     subset_sum ← Tuple find_closest_subset_sum(class_frequencies, target_sum)
8:     test_clusters ← List get_clusters(subset_sum, clusters)
9:     difference ← Int sum(target_sum - subset_sum)    ▷ Outer sum over the tuple
10:    split_shortlist ← Set split_shortlist ∪ {($c$, difference, test_clusters) }
11: end for
12: final_test_clusters ← List get_best_test_clusters(split_shortlist)
                ▷ Best means least clusters with the lowest difference to target_sum
13: test_examples ← Set get_examples(final_test_clusters, $X$)
14: while class_frequencies(test_clusters) ≤ target_sum do    ▷ No exact solution
15:     test_examples ← Set test_examples ∪ { example from another cluster}
16: end while
17: train_examples ← Set $X$ \ test_examples
18: return train_examples, test_examples
```

3.2.2.2 Algorithm 2: Closest-Split

In contrast to the subset-sum-split, the closest-split aims to put as much distance as possible between the train set clusters and the test set clusters. This leads to an even bigger under-representation of parts of the latent space in the training set.

Once the clusters have been determined, their centroids are calculated. The cluster that lies farthest away from all the other clusters is identified and added to the test set. Cosine similarity between cluster centroids is used as the distance measure. If the size of the farthest cluster exceeds the target test set size, the next farthest cluster is taken instead. Then, successively, the nearest cluster to any of the clusters already in the test set is added as long as the size of the test set does not exceed the target size. In essence, this is *nearest neighbour* clustering with the cluster centroids. When this nearest-neighbour clustering is finished, individual examples that are closest to one of the test set centroids are added to the test set until the target size is reached. As with subset-sum-clustering, the algorithm is performed for $k = 3$ to $k = 50$ clusters, and

Algorithm 2 Closest-Split

```
1: Input: $n$ latent representations Set $X = \{x_i\}_{i=1}^{n}$
2: target_sum ← Tuple ($n \cdot$ hate ratio $\cdot$ test ratio, $n \cdot$ noHate ratio $\cdot$ test ratio)
3: split_shortlist ← Set
4: for $c = 2$ to Maximum Number of Clusters do
5:     clusters, centres ← List, List kmeans(X, c)
6:     class_frequencies ← List get_class_frequencies(clusters)        ▷ List of tuples
7:     meta_centre ← Tuple mean(centres)
8:     dist_to_meta_centre ← List cosine_sim(centre, meta_centre) for all centres
9:     farthest_cluster ← Cluster $\arg\max_{\text{cluster}}$ dist_to_meta_centre
10:    while class_frequencies(farthest_cluster) > target_sum do
11:        candidate ← Cluster get_next_farthest_cluster(dist_to_meta_centre)
12:    end while
13:    test_clusters ← Set
14:    while class_frequencies(test_clusters ∪ {candidate}) ≤ target_sum do
15:        test_clusters ← Set test_clusters ∪ {candidate}
16:        clusters ← Set clusters \ {candidate}
17:        candidate ← Cluster get_closest_cluster(clusters, test_clusters)
18:    end while
19:    split_shortlist ← Set split_shortlist ∪ test_clusters
20: end for
21: final_test_clusters ← Setget_best_test_clusters(split_shortlist)
                 ▷ Best means least clusters with the lowest difference to target_sum
22: test_examples ← Set get_examples(final_test_clusters, $X$)
23: while class_frequencies(test_clusters) ≤ target_sum do        ▷ No exact solution
24:     test_examples ← Set test_examples ∪ { closest example from correct class}
                             ▷ distance = cosine_sim(·, mean(test_examples))
25: end while
26: train_examples ← Set$X$ \ test_examples
27: return train_examples, test_examples
```

the data split with the fewest individual examples additionally added is chosen. The algorithm is illustrated in Algorithm 2.

In contrast to the subset-sum-split, the closest-split has a polynomial (quadratic) runtime and is therefore significantly faster.

3.2.3 Models Used to Create and Evaluate the Data Splits

This work uses transformer language models (Vaswani et al., 2017; Devlin et al., 2019) to obtain hidden representations and to test the new train-test split. In general, transformers are fully-connected neural networks that have been enhanced with an attention mechanism (Vaswani et al., 2017). For classification tasks, encoder-only models are used. In encoder-only transformers, the attention mechanism is self-attention, which, roughly speaking, enhances the representation of each input word with information about its context. Transformers are pretrained in an unsupervised manner and are then finetuned in a supervised manner on a specific task. Specifically, this work uses BERT models (Devlin et al., 2019). BERT models are bidirectional encoders that are pretrained on two tasks, *masked language modelling* (MLM) and *next sentence prediction* (NSP). They obtained state-of-the-art performance on a variety of tasks (Devlin et al., 2019), and even though they are now outperformed by other models, they are still widely used and analysed. The following four variants are used in this work:

- **Bert-Base(-Cased)** is the smaller of the two standard BERT models as proposed in Devlin et al. (2019). It was pretrained on the BooksCorpus (Zhu et al., 2015) and English Wikipedia.
- **Bert-Medium** (Turc et al., 2019; Bhargava, Drozd, and Rogers, 2021) is a more compact BERT-model that has less than half as many parameters as BERT without a big performance drop.
- **HateBert** (Caselli et al., 2021) is a Bert-Base-Uncased model that was further pretrained on abusive Reddit data using the MLM objective. As a result, it performs slightly better on three hate speech datasets (Basile et al., 2019; Caselli et al., 2020; Zampieri et al., 2020) than a normal BERT model.
- **RoBERTa** (Liu et al., 2019) has a similar architecture to BERT, trained with bigger batches and more than ten times as much data. Moreover, the pretraining tasks are slightly modified: Masking is performed dynamically; different words are masked every time an input sequence is fed into the model. The NSP objective is removed.

A detailed description of the models' sizes and numbers of parameters can be found in Table A.1 in Appendix A. The hyperparameters used for finetuning are listed in Table A.2. Generally, the hyperparameters were adopted from the HateBert finetuned models (Caselli et al., 2021), but due to computational restrictions, the models had to be trained with reduced batch sizes. To compensate for this, models were trained with more epochs with the option of early stopping.

3.3 Experimental Setup

This section presents the experiments used to develop and evaluate the data-splitting method proposed in Section 3.2. The experiments are divided into three phases: First, experiments that serve as baselines for further experiments and allow comparison with the language model results from the literature on the standard data split; second, several different dataset splitting methods are developed and tested on the development dataset (presented in Section 3.1.1). Third, the method that works best is then applied to a new, external test dataset (presented in Section 3.1.2) and evaluated to ensure that the method does not only work for the development dataset.

3.3.1 Baselines

The first experiments obtain a baseline. For the test dataset (Mathew et al., 2020), the models are trained and evaluated on the standard split as proposed for the dataset. Since no standard split is given on the development dataset (Qian et al., 2019), a random split is used to compare the performance of the models in this work to the literature so as to ensure that the models act as expected on the standard (or random) split. Another random baseline is obtained with 90% of the data (10% are set aside as independent test data) in order to compare the new splits against a fair random baseline since one would expect a small deterioration of performance when training the model with fewer data.

3.3.2 Development Dataset - Experiments

The experiments on the development dataset include an extensive hyperparameter search and testing of design choices to determine the best dataset-splitting method, i.e. the method that produces the most challenging data split for evaluating generalisation.

Specifically, the data-splitting framework (as introduced in Section 3.2) comes with several different choices for each component of the algorithm:

- **Obtaining the hidden representations:** Hidden representations are obtained after finetuning the language models on the dataset. Four different numbers of hidden dimensions are considered: 10, 50, 200 and the full set of dimensions as internally used by the model. The dimensionality reduction is either performed by UMAP or achieved through the addition of a new bottleneck layer before finetuning the model.
- **Assigning clusters to the train or test set:** Both proposed cluster assignment algorithms (subset-sum-split and closest-split) are tested.

For each of the above (14) combinations, four different language models (presented in Section 3.2.3) are used to extract hidden representations, which are in turn clustered with three different cluster seeds. When finetuning language models on the new data splits, every model is run with three different initialisations for the classification head.

The splitting approach so far has relied on task-specific finetuned representation. To analyse whether task-specific representations are needed to create challenging data splits for the task, the dataset is also split based on hidden representations obtained from the pretrained models. Pretrained language model embeddings capture general, task-agnostic information about the inputs. Therefore, the experiments described above are repeated with the pretrained hidden representations. Note that a bottleneck cannot be used for dimensionality reduction here, as it would require further (pre-)training.

3.3.3 Test Dataset - Experiments

Having conducted the experiments on the development dataset, the design choices for the splitting method are narrowed down for the test dataset, by keeping the most successful parameters: only full- and 50-dimensional representations are considered further. Finally, two additional sets of experiments are conducted using only the best-performing combination – bottleneck-50 closest-split:

- **Independent Test Set Performance:** Decreased model performance on the new data split could either mean that the new test set is particularly challenging or that the new train set is no longer informative enough to learn the task. To rule out the latter case, the finetuned models are tested on the *independent test data* that was set aside earlier.
- **Cross-Model Performance:** Obtaining a dataset split for individual models is costly. Therefore it is also examined, whether a split obtained by the hidden representations of a specific model is also challenging for other language models.

As for the development dataset, all experiments are conducted with four different language models and three different classification head initialisations.

3.3.4 Evaluation Metrics

Accuracy, hate F1-score (macro F1-score for the test dataset) and Roc Auc score are reported in all of the experiments. This is in line with the associated dataset papers (Qian et al., 2019; Mathew et al., 2020). Finally, the accuracy on the validation set is recorded to assess the model's ability to generalise on part of the train set.

Chapter 4

Results

This section presents the experimental results of this work. The goal of all the experiments is to find a dataset split that is more challenging than a random data split to test the model for robust generalisation. Different splitting methods are explored on a development dataset (Section 4.1) and the most promising methods are then also evaluated on a test dataset (Section 4.2).

4.1 Results for the Development Dataset

The development dataset (Qian et al., 2019) is a hate speech dataset that consists of two classes, *hate* and *noHate*.

4.1.1 Baseline Results Using Random Splits

A series of baseline experiments are conducted to establish that the models used in this work perform as expected based on the relevant literature. Qian et al. (2019) do not present a standard data split when introducing their dataset. Instead, they report results on a random split for an SVM model and an RNN model (Cho et al., 2014). To check the performance of the models in this work, the data was split with three different seeds and the average performance on these splits is reported. Note that in addition to these baselines obtained using the whole dataset, baselines are also obtained using only 90% of the data. This is done to provide a more comparable baseline for the proposed split methods, which also split 90% of the data (as introduced in Section 3.2.1). The SVM and RNN of Qian et al. (2019) reach F1 scores of 75.5 and 77.5 respectively on the dataset. Unsurprisingly, the transformer language models BERT (Devlin et al., 2019),

model	valid acc.	test acc.	f1	roc auc
SVM (Qian et al., 2019)	–	–	75.7	81.1
RNN (Qian et al., 2019)	–	–	77.5	79.4
BERT-base (100%)	94.6 ± 0.21	91.55 ± 0.13	82.24 ± 0.34	94.32 ± 0.27
BERT-base (90%)	91.69 ± 0.07	91.25 ± 0.11	81.96 ± 0.5	94.35 ± 0.18
BERT-medium (100%)	94.3 ± 0.23	91.63 ± 0.2	82.27 ± 0.45	94.35 ± 0.17
BERT-medium (90%)	91.84 ± 0.07	91.2 ± 0.15	81.58 ± 0.66	94.45 ± 0.16
HateBert (100%)	94.12 ± 0.06	91.87 ± 0.16	82.72 ± 0.38	95.06 ± 0.07
HateBert (90%)	92.02 ± 0.07	91.51 ± 0.13	82.34 ± 0.59	95.14 ± 0.15
RoBERTa (100%)	94.4 ± 0.12	91.67 ± 0.2	82.5 ± 0.49	94.63 ± 0.15
RoBERTa (90%)	91.8 ± 0.09	91.37 ± 0.16	82.15 ± 0.61	94.32 ± 0.21

Table 4.1: Results for the development dataset on random splits using 100% and 90% of the data. Random splits are generated using three different seeds and models are trained with three initialisation seeds, mean and standard errors are reported.

BERT-Medium (Turc et al., 2019), HateBert (Caselli et al., 2021) and RoBERTa (Liu et al., 2019) perform significantly better with F1 scores of around 82 and roc auc scores also up to 14 points higher than the SVM and RNN. Perhaps more surprising is that HateBert only marginally outperforms BERT-base, even though it has been especially pretrained for hate speech detection. Training the model with only 90% of the data leads to a small decrease of around 0.4 points.

4.1.2 Subset-Sum-Splits are More Challenging

The subset-sum-split (introduced in Section 3.2.2.1) puts clusters based on hidden representations in either the train or the test set. The hypothesis is that an under-representation of parts of the embedding space leads to a challenging data split to test generalisation. Indeed, the empirical results show significant performance drops when training language models on this split. The performance differences compared to using a random split are shown in Fig. 4.1.

On most of the subset-sum-splits the performance drops to an accuracy between 75% and 85% in comparison to the random split baseline of 91%. However, some data splits do not seem to be more challenging than a random split; with one even showing a performance increase to 95% accuracy (RoBERTa-umap-10 split).

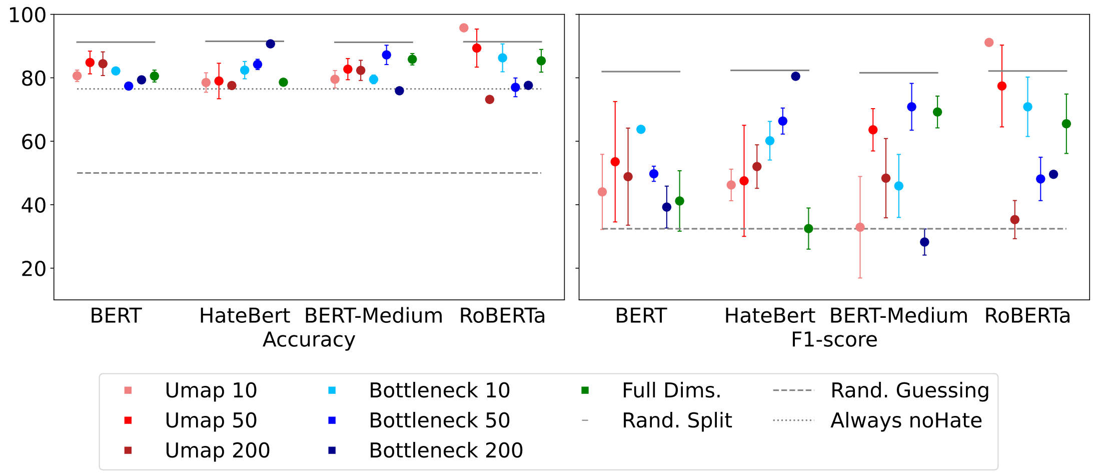


Figure 4.1: Performance of language models trained on the subset-sum-split of the development dataset. Random split performance, indicated by the solid horizontal lines, is used as a baseline. The error bars show the standard error between cluster seeds.

In terms of F1 scores for the hate class, the performance drops drastically for most of the splits. The models opt for the hate class very rarely, leading to lower hate recall rates and decreased overall performance. Some data splits, for example the HateBert-full-dimensional split, even lead to a performance that is on par with a random guessing baseline. Splits obtained with lower dimensional representations (lighter colours in Fig. 4.1) generally lead to greater decreases in performance. Moreover, obtaining the dimensionality reduction with the help of UMAP seems to help further decrease performance. On the other hand, decreasing the dimensionality of the hidden representations with UMAP also leads to higher variance between cluster seeds, as the error bars indicate.

The language models all seem to behave similarly on the harder test datasets, with RoBERTa models reacting a little more robustly than the other models.

4.1.3 Closest-Splits are More Challenging than Subset-Sum-Splits

The closest-split aims to put even more distance between the train and test sets by sorting the clusters globally. In this way, the split under-represents one large area of the latent space in the training set. The underlying hypothesis is that this leads to even greater and more consistent performance drops. And indeed, this is seen in the results. The decreases in performance compared to a random split are depicted in Fig. 4.2.

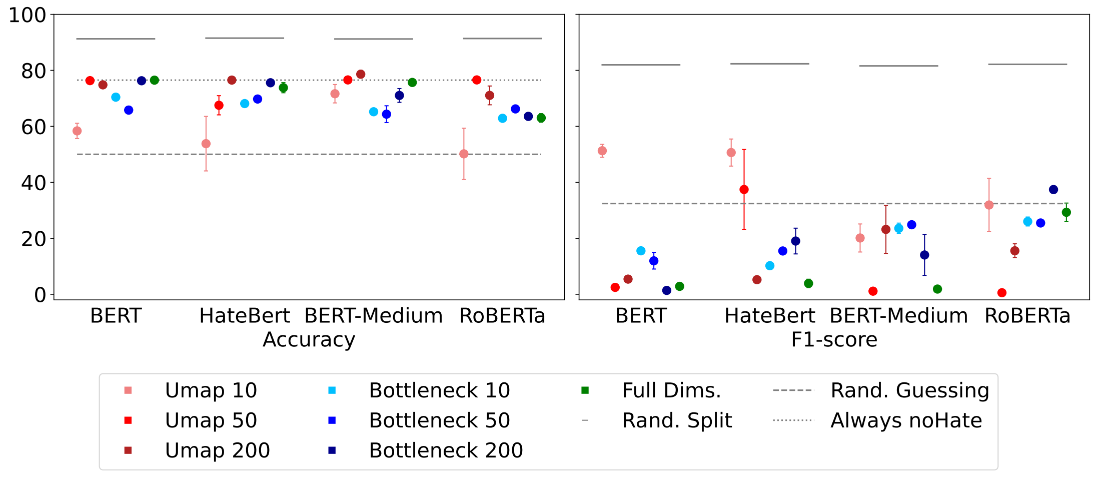


Figure 4.2: Performance of language models trained on the closest-split of the development dataset. Random split performance, indicated by the solid horizontal lines, is used as a baseline. The error bars show the standard error between cluster seeds.

For most of the closest-splits, performance drops from the random split accuracy of 91% to an accuracy between 65% to 75%. This might not seem too low, however, since the dataset is imbalanced, predicting only the noHate class already leads to an accuracy of 76.5% (dotted line in the plot). Looking at the F1 scores, it seems that this is indeed what happens. The F1 scores are mostly between 0 and 25, indicating a large discrepancy between the performance on the different classes. In contrast to the subset-sum-split, the closest-split leads to a consistent decrease across all models and all latent space representations, with not a single split leading to increased scores. Moreover, there is also less variance between cluster seeds, with the exception of the data splits obtained from the UMAP-reduced latent representations with 10 and 50 dimensions.

4.1.4 Task-Specific Embeddings are Crucial for the Data Split

The previous two sections have shown that splitting the data in a model's internal embedding space leads to decreased test set performance. However, it is not clear, whether task-specific embeddings are needed to create challenging data splits for the task, or whether using more general pretrained embeddings would lead to similar drops in performance. Pretrained embeddings capture information about language in general, and the language model that produces these pretrained embeddings has not been exposed to the task. If model performance drops significantly less when splitting based on the pretrained embeddings, then task-specific (finetuned) embeddings are needed to produce more challenging data splits.

In fact, for the (pretrained) closest-split, the models show almost no decrease in model performance. When the full-dimensional data split is used, BERT-base and BERT-medium show a decrease in accuracy of 7%, while HateBert and RoBERTa do not show a decrease. When using UMAP dimensionality reduction, none of the models shows a performance decrease at all. For the (pretrained) subset-sum-split, models show either no or only a really small decrease in performance (−3% accuracy). This indicates that task-specific features are crucial for creating challenging splits with the proposed data-splitting methods.

Detailed results can be found in Table B.2 and Table B.1 in Appendix B.

4.2 Results for the Test Dataset

Having tested different splitting methods on the development dataset, the question arises whether these findings are specific to the development dataset or perhaps even specific to binary classification. The results in this section reveal that the same tendencies can be replicated on another dataset. The test dataset from (Mathew et al., 2020), which includes three classes (*hate*, *offensive*, *normal*), is examined. In the previous section, experiments were conducted with a total of 14 splits. Based on those results, the following six splits are selected for the test set: the closest-split and the subset-sum-split for each of umap-50, bottleneck-50 and full-dimensional.

4.2.1 Baseline Results Using Random Splits

The test dataset (Mathew et al., 2020) comes with a standard data split and a BERT-base baseline. Therefore, this work also evaluates the four language models that are used for the experiments on this standard split. As before, a random split baseline on 90% of the data is also calculated for comparison. The baselines are presented in Table 4.2. Mathew et al. (2020) report a test accuracy of 69% for a BERT-base-uncased model. The language models in this work perform similarly: the BERT-base-cased model is 0.6% worse than reported, and HateBert and RoBERTa perform at around 68%. However, the smaller Bert-Medium model only reaches 64.6% accuracy. Taking into account a smaller batch size during training due to computational restrictions and the standard error on the results, the baselines on the standard split are reasonable in comparison to

split	model	valid acc	test acc	f1	roc auc
stand.	BERT-base *	–	69.0	67.4	84.3
stand.	BERT-base	67.45 ± 0.36	68.38 ± 0.35	66.06 ± 0.44	84.04 ± 0.19
	BERT-med.	63.93 ± 1.2	64.58 ± 0.99	62.32 ± 1.45	81.79 ± 0.96
	HateBert	68.12 ± 0.16	68.0 ± 0.37	65.97 ± 0.36	84.06 ± 0.23
	RoBERTa	67.32 ± 0.3	67.83 ± 0.42	65.98 ± 0.26	84.03 ± 0.27
rand.	BERT-base	67.66 ± 0.31	68.25 ± 0.28	66.0 ± 0.36	83.6 ± 0.14
	BERT-med.	62.46 ± 0.49	62.85 ± 0.42	60.18 ± 0.42	79.98 ± 0.34
	HateBert	67.91 ± 0.32	68.51 ± 0.28	66.25 ± 0.35	84.46 ± 0.15
	RoBERTa	66.45 ± 0.51	66.4 ± 0.56	64.1 ± 0.9	82.19 ± 0.61

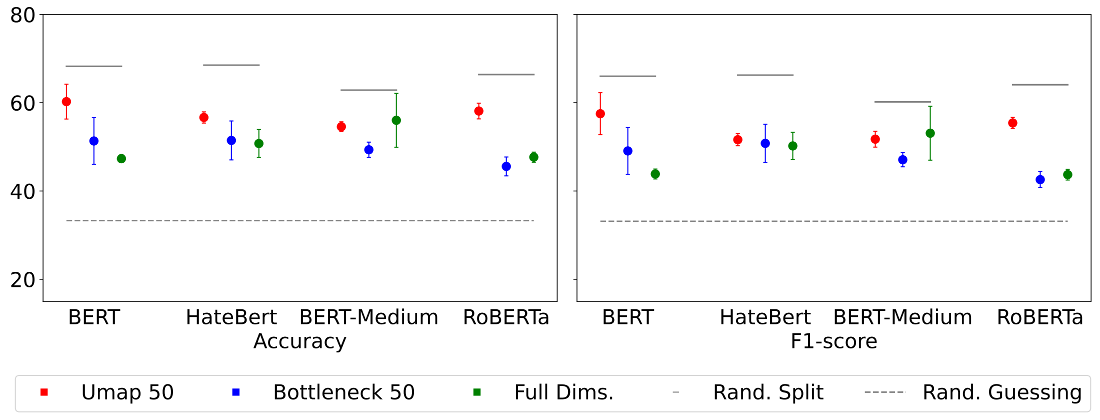
Table 4.2: Results for the test dataset on the standard (stand.) split and on random (rand.) splits using 90% of the data. Random splits are generated using three different seeds and models are trained with three initialisation seeds, mean and standard errors are reported. Results marked with * are taken from Mathew et al. (2020).

the reported baseline. Results on the random split (averaged over three different random seeds) are slightly lower in comparison to the standard split, which is not surprising given that the random split uses only 90% of the data.

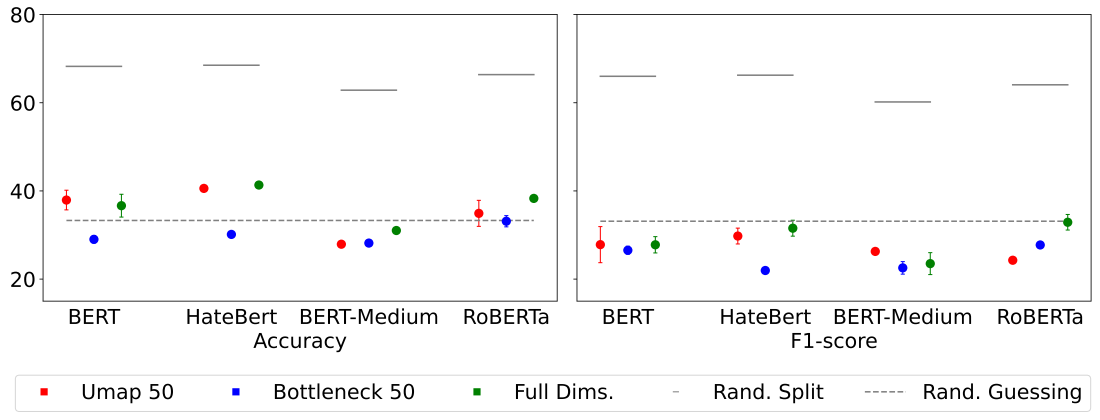
4.2.2 The Data Split is also Effective on the Test Dataset

The results for the subset-sum-split and closest-split on the test dataset show the same trends as on the development dataset, supporting the hypothesis that the splitting method works independently of the dataset. As before, the closest-split leads to greater and more consistent decreases in performance, also with less variance across random cluster seeds. The results are shown in Fig. 4.3.

For the subset-sum-split (Fig. 4.3a), both the accuracy and the F1 scores of the tested language models drop around 15 points in comparison to training on a random split. In contrast to the development set, there is no major difference between F1 and accuracy scores. This is most likely due to the fact that the three classes in the test dataset are rather balanced. Detailed results for the subset-sum-split can be found in Table B.3 in Appendix B.



(a) Subset-sum-split



(b) Closest-split

Figure 4.3: Performance of language models trained on the subset-sum-split and closest-split of the test dataset. Random split performance, indicated by the solid horizontal lines, is used as a baseline. The error bars show the standard error between cluster seeds.

Training models on the closest-split leads to more drastic performance drops (Fig. 4.3b). The accuracies of the language models drop to around 35%, with F1 scores dropping even lower. The most affected by the data split is the BERT-medium model with an accuracy of 27.74 and an F1 score of 21.56 for the bottleneck-50 split.

Generally, the splits that use a bottleneck to reduce the dimension of the latent representations (blue in Fig. 4.3) lead to the most challenging data splits. Therefore, further experiments with independent test data and cross model splits are conducted with the closest-splits obtained by this setting.

model	split	valid acc	test acc	f1	roc auc
BERT-base	random	67.66 ± 0.31	68.25 ± 0.28	66.0 ± 0.36	83.6 ± 0.14
	closest	72.44 ± 0.75	29.73 ± 0.41	27.14 ± 0.17	46.53 ± 0.47
	ind. test	–	67.05 ± 1.05	64.83 ± 1.32	82.61 ± 0.89
BERT-med.	random	62.46 ± 0.49	62.85 ± 0.42	60.18 ± 0.42	79.98 ± 0.34
	closest	68.27 ± 0.32	27.77 ± 0.56	21.56 ± 0.34	45.79 ± 0.21
	ind. test	–	63.2 ± 0.54	60.77 ± 0.81	80.22 ± 0.32
HateBert	random	67.91 ± 0.32	68.51 ± 0.28	66.25 ± 0.35	84.46 ± 0.15
	closest	72.1 ± 0.33	30.19 ± 0.61	21.66 ± 0.49	49.8 ± 1.0
	ind. test	–	67.05 ± 1.05	64.83 ± 1.32	82.61 ± 0.89
RoBERTa	random	66.45 ± 0.51	66.4 ± 0.56	64.1 ± 0.9	82.19 ± 0.61
	closest	70.86 ± 0.33	34.22 ± 1.62	29.5 ± 1.29	51.48 ± 3.94
	ind. test	–	67.24 ± 0.28	65.4 ± 0.38	82.95 ± 0.25

Table 4.3: Results on the test dataset. The models are trained on the closest-split train data and tested on the closest-split test data (*closest*) and on an independent test set (*ind. test*). These results are compared to models that are trained and tested on a random split (*random*). The mean and standard error are calculated across different model initialisations and cluster seeds.

4.2.3 Closest-Split Models Perform Well on Independent Data

The splitting methods proposed in this work lead to decreased performance, showing that the language models are not able to generalise as robustly as on random data splits. The question remains whether the test set obtained by the new splitting methods is harder or whether the new splitting method leads to very simple or perhaps even incomplete training sets, thereby preventing the models from learning the task. The following results strengthen the hypothesis that the test set is indeed harder.

One measure to examine whether the models learn something during training is to examine the performance on the validation set (a part of the train set that is used to estimate a model's ability to generalise during training). The validation accuracy for the models trained on closest-split and subset-split is for most splits around 5 points higher than the accuracy on the validation set of the random data split. This shows that the models are indeed able to learn during training. Moreover, the even increased validation score hints towards the assumption that the test set is indeed more difficult and using the difficult examples only for testing makes the train and validation sets easier. Validation accuracies for models trained on the closest-split with a bottleneck of dimension 50 are reported in Table 4.3, for all other splits the reader is referred to Table B.3 and Table B.4 in Appendix B.

A second measure to examine whether the model failed to learn the task is testing it on the *independent test data*. Specifically, the models that were trained on the closest-split are tested on data that was not used to create the data split. This data was set aside earlier and has never been seen by a model during training. Results for these experiments are reported in Table 4.3. All models that have been trained on the closest-split achieve similar accuracy on the *independent test data* as the models trained and tested on random data, with BERT and HateBERT achieving a slightly lower performance than on the random split, and BERT-medium and RoBERTA a slightly higher performance. This also supports the hypothesis that the data-splitting method does not split the data in a way that the model is not able to learn from the training data. On the contrary, the model trained on the closest-split data performs just as well on the independent test data.

4.2.4 Closest-Splits are Challenging for Other Models

Having shown that closest-splits indeed find a harder test set that leads to decreased performance on the task without hindering the model in learning the task, this section addresses the question of whether the method only works as a model-specific approach, or whether a split that uses representations from one model is also challenging for other models. Since the ultimate goal of the new data split is to improve the evaluation of generalisation, data splits that are challenging for more than one model are more practical because 1) the data-splitting pipeline only needs to be carried out with one model and 2) multiple models can be assessed and compared with the same split.

The results of the cross-model evaluations can be seen in Table 4.4. They show that data splits obtained on the hidden representations of one model are indeed also challenging for other models, although not as challenging as the personalised data splits. For example, a closest-split with BERT-base hidden representations leads to a performance drop of 38.5 accuracy points for the same BERT-base model (in comparison to the random split performance of 68.25%). But accuracy still drops by 30%, 34.7% and 30.7% respectively when BERT-medium, HateBert and RoBERTa models are trained on this data split. The same trend can be observed for the F1- and roc auc scores.

model	split	valid acc	test acc	f1	roc auc
BERT-base	random	67.66 ± 0.31	68.25 ± 0.28	66.0 ± 0.36	83.6 ± 0.14
	BERT-b.	**72.44 ± 0.75**	**29.73 ± 0.41**	**27.14 ± 0.17**	**46.53 ± 0.47**
	BERT-m.	72.09 ± 0.14	38.57 ± 0.33	36.59 ± 0.54	58.42 ± 0.25
	HateB.	72.12 ± 0.32	33.51 ± 0.32	27.73 ± 0.64	52.34 ± 0.1
	RoB.	70.94 ± 0.2	37.52 ± 0.21	31.07 ± 0.34	59.17 ± 0.74
BERT-medium	random	62.46 ± 0.49	62.85 ± 0.42	60.18 ± 0.42	79.98 ± 0.34
	BERT-m.	**68.27 ± 0.32**	**27.77 ± 0.56**	**21.56 ± 0.34**	**45.79 ± 0.21**
	BERT-b.	66.53 ± 0.22	30.56 ± 0.38	28.19 ± 0.38	48.8 ± 0.16
	HateB.	66.55 ± 0.8	31.75 ± 0.54	26.3 ± 0.26	50.43 ± 0.11
	RoB.	66.15 ± 0.64	35.13 ± 0.32	29.73 ± 0.58	57.66 ± 0.08
Hate-Bert	random	67.91 ± 0.32	68.51 ± 0.28	66.25 ± 0.35	84.46 ± 0.15
	HateB.	**72.1 ± 0.33**	**30.19 ± 0.61**	**21.66 ± 0.49**	**49.8 ± 1.0**
	BERT-b.	71.27 ± 0.34	35.24 ± 0.14	32.08 ± 0.26	53.11 ± 0.26
	BERT-m.	71.52 ± 0.35	40.54 ± 0.63	38.53 ± 0.88	59.0 ± 0.59
	RoB.	70.23 ± 0.49	38.75 ± 0.32	32.41 ± 0.55	60.83 ± 0.12
Ro-BERTa	random	66.45 ± 0.51	66.4 ± 0.56	64.1 ± 0.9	82.19 ± 0.61
	RoB.	**70.86 ± 0.33**	**34.22 ± 1.62**	**29.5 ± 1.29**	**51.48 ± 3.94**
	BERT-b.	69.77 ± 0.75	33.16 ± 0.52	31.04 ± 0.62	48.75 ± 0.4
	BERT-m.	70.15 ± 1.0	38.21 ± 0.64	36.96 ± 0.33	57.3 ± 1.49
	HateB.	73.04 ± 0.18	34.82 ± 0.06	30.02 ± 0.3	51.48 ± 0.27

Table 4.4: Results for the test dataset on a bottleneck-50 closest-split. Comparison of models trained on the data split obtained with their respective hidden representations (in bold) and on data splits obtained from hidden representations of other models. The mean and standard error are calculated across different model initialisations and cluster seeds.

Generally, the BERT-medium split seems to be the hardest with an average accuracy of 31.3% across the four models, followed by BERT-base (34.83%), RoBERTa (35.10%) and HateBert (36.18%). But even the least challenging split by HateBert leads to a significant performance drop across all models, making closest-split an interesting approach for evaluating the generalisation ability of models.

Chapter 5

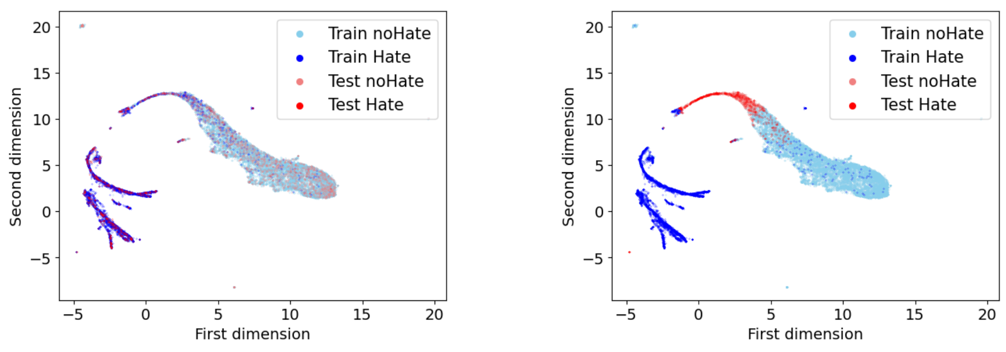
Analysis

The performance of models deteriorates heavily when the closest-split is used to split the data, making the closest-split an interesting method for evaluating o.o.d. generalisation. This section analyses the splits that it generates; first by looking at the hidden representations, and then at the properties of the resulting train and test sets. Since the closest-split with hidden representations obtained with a bottleneck of dimension $d = 50$ leads to the biggest performance decreases, this setting is used in the analysis below.

5.1 Analysis of the Latent Representations

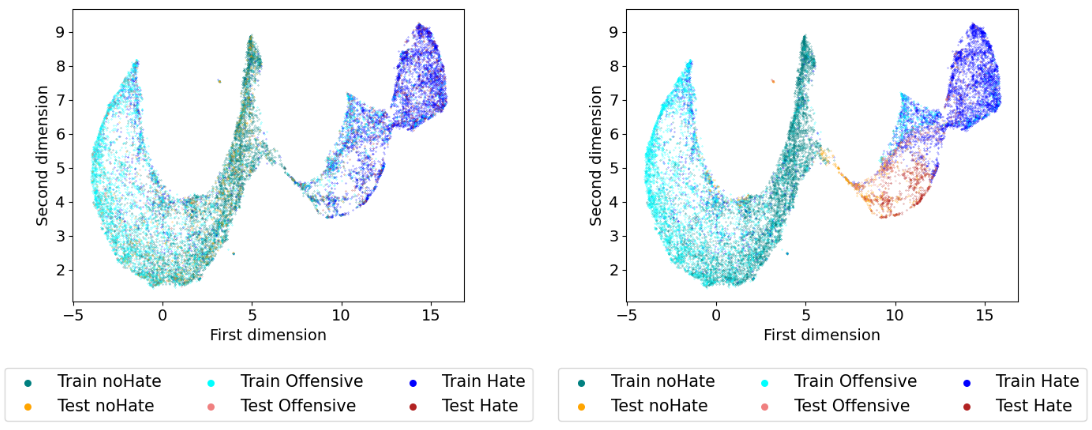
The closest-split uses hidden representations of a language model and puts clusters of these into either the train or the test set. This is done by placing clusters that are spatially close into the same set. While the clustering is conducted on high-dimensional representations, an illustration reduced to two dimensions can give an intuition of the data split. Dimensionality reduction to two dimensions is performed by UMAP (McInnes, Healy, and Melville, 2020).

Fig. 5.1 shows such an illustration of the hidden representations of a BERT model, trained on the development dataset on the binary classification task. The hidden representations of the model seem to define a decision boundary between the dark hate class on the left and the lighter noHate class on the right. Only at the top of the noHate cluster there are also some hate examples. While in a random split, the test examples are present in all regions of the hidden representations, the closest-split now assigns examples from exactly this mixed region to the test set (the closest-split uses clusters that are the farthest away in space from the centre of the hidden representations). While



(a) Random Split (b) Closest-split

Figure 5.1: BERT hidden representations for binary classification (development data)

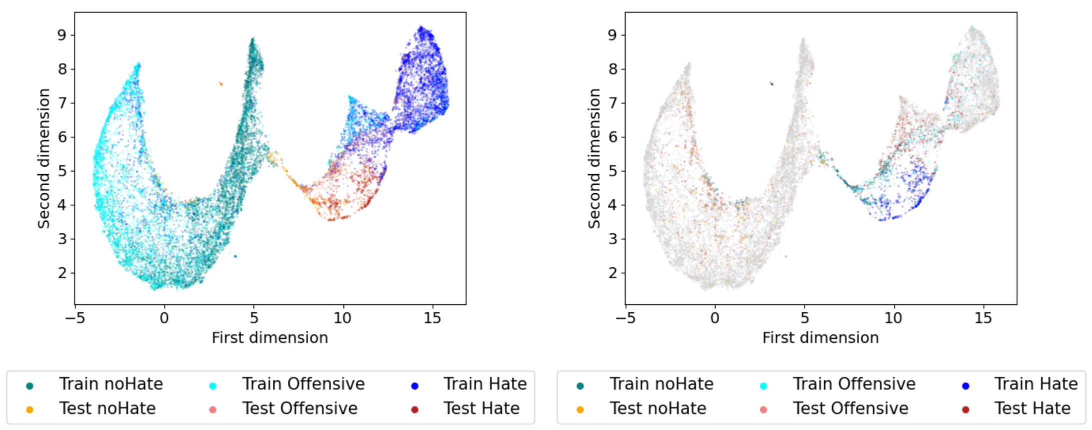


(a) Random Split (b) Closest-Split

Figure 5.2: RoBERTa hidden representations for tertiary classification (test data)

the decision boundary appears to be quite clear on the train set, this is not the case in the test set anymore. So this might be a reason, why models struggle to predict the correct class in the test set (with an F1 score of 14.16 for the same model) but are confident for examples in the train and validation sets.

The same phenomenon holds for the test dataset, which poses a three-way classification task. Fig. 5.2 shows the hidden representations of a RoBERTa model trained on the test dataset. Again, a decision boundary can be observed, with mostly offensive examples on the left, normal examples in the middle and hate examples on the right. The decision boundaries are not as exact as on the development dataset. This is reflected in the lower performance on random splits: random split accuracy is about 90% on the development dataset, but only 68% on the test dataset. Based on Fig. 5.2, the closest-split picks a pocket of (mixed) examples between the normal (dark blue) and hate (dark green)



(a) RoBERTa closest-split (b) BERT closest-split

Figure 5.3: RoBERTa hidden representations, shown here with two different splits (RoBERTa closest-split and BERT closest-split). The grey area in the right plot depicts the data examples that are in the same set as in the RoBERTa split.

regions to be the test set. This is then mirrored in the F1 scores of the different classes. The test example for the hate class lie closest to the corresponding region and the F1 score is the highest at 47.0. Similarly, for the normal class, the F1 score is relatively high at 38.28. The offensive class, with test examples far away from its corresponding region, only has an F1 score of 11.88.

Note, that the plots look similar for the other language models, whose hidden representations are not presented here.

Do these properties also hold, when training a language model on a split obtained using another language model? Section 4.2.4 has shown a decreased performance in such a scenario, although the decrease was not as big as when obtaining the split with the same model. Fig. 5.3 shows an illustration of the hidden representations of a RoBERTa model, coloured according to a data split obtained by RoBERTa hidden representations (on the left) and by BERT hidden representations (on the right). The intersection between both splits is depicted in grey. The phenomenon described above is visible in both scenarios, albeit clearly weakened on the right: The test set still contains examples in the same region, but not only. Examples that are close in space for BERT hidden representations are not necessarily close in space for the RoBERTa hidden representations, and therefore some examples of the test set are distributed over the whole space. This reflects in the results when training the models, the RoBERTa model trained on the RoBERTa split reaches an F1 score of 29.5, whereas the RoBERTa model trained on the BERT split reaches a slightly higher F1 score of 31.04.

	BERT	BERT-Medium	HateBert	RoBERTa
BERT	94.06 ± 4.04			
Bert-Medium	32.95 ± 0.89	95.22 ± 3.01		
HateBert	49.28 ± 2.3	26.23 ± 0.91	93.14 ± 2.37	
RoBERTa	32.55 ± 11.12	25.39 ± 12.37	34.81 ± 13.64	17.54 ± 12.4

Table 5.1: Test set overlap between closest-splits based on hidden representations of different models. Values are the percentage overlap between test set examples; the mean and standard deviation are taken over multiple cluster seeds. The diagonal represents the test set overlap for the same model across different cluster seeds.

More concretely, Table 5.1 presents the overlap of test set examples between splits using the latent representations of different models. The BERT and HateBert test sets have a high overlap of almost 50%, whereas the other model combinations have a lower overlap of between 25% – 35%. While most models have high within-model overlap across different cluster seeds, above 90%, the RoBERTa splits do not. This means that even though the data-splitting procedure can end up with very different test sets, the performance decrease across splits is consistent (as showed in Section 4.2).

5.2 Analysis of the Properties of the Data Split

Having taken a closer look at the dataset split with respect to the latent spaces of the internal model representations, this section now analyses the data split with respect to the actual text inputs. Do the test examples substantially differ from the train examples in terms of length or vocabulary or some other measure? And if there is a noticeable difference on the text level, is it the same for both datasets? Section 5.2.1 presents a range of task-agnostic analysis methods for data splits and Section 5.2.2 contains analysis methods specifically designed for hate speech. Section 5.2.3 presents the results of the analysis.

5.2.1 Methodology: Task-Agnostic Analysis

Word Overlap: A straightforward way to compare the train and test sets is to examine how similar they are in terms of word overlap. If a lot of words appear only in the test set and the model has not seen them during training, then a model might have difficulties classifying respective examples. The overlap is calculated following the word overlap algorithm in Elangovan, He, and Verspoor (2021). More details on this algorithm can be found in Appendix C.1.

Rare Word Count: It might be particularly challenging for a model if rare words appear in the test set as such words might also not have been abundant during pretraining, let alone during finetuning. To examine, whether more rare words appear in the test set than in the train set, the number of rare words is extracted, following the definition of Godbole and Jia (2022). Appendix C.1 gives details about this definition.

Sentence Length: For natural language generation tasks, it is known that longer sentences are harder for language models (Varis and Bojar, 2021). Therefore, the average sentence length of the train and test sets will be analysed.

5.2.2 Methodology: Task-Specific Analysis

Apart from the task-agnostic analysis methods above, hate-speech-specific analyses are also performed. This is particularly important in this work, as the split itself is obtained by clustering task-specific representations.

Targets of Hate Speech: This method aims to analyse the different targets of hate speech. For the test dataset (Mathew et al., 2020), these targets are annotated as explained in Section 3.1. If the model has not been exposed to a specific target group during training, this might lead to difficulties detecting hate speech against this group in the test set. To this end, the difference between the target distribution in the train and test sets is analysed. The Kullback-Leibler Divergence (Kullback and Leibler, 1951, KL Divergence) is calculated for each target in the dataset and then the average is taken over all targets, weighted by the occurrence of the target in the dataset. Since there is no upper bound for the KL Divergence, it is scaled to be between 0 and 1 by the function

$$f(x) = 1 - e^{-x}. \tag{5.1}$$

Additionally, the number of targets that are under-represented in the train set i.e. have less than 50% of their occurrences in the train set, is computed. Targets that occur in less than 3% of the data set are excluded.

For the development dataset (Qian et al., 2019), no such target groups are annotated. However, it is known which posts belonged to the same conversation. Thus, under the assumption that a conversation has similar hate targets, the split of conversations between the train and test sets is analysed. Note, that there are significantly more conversations in the development dataset than targets in the test dataset.

Hate Keywords: The development dataset and the test dataset have been created by filtering posts based on hate keywords by simply string-matching the posts with the keywords. These keywords can be understood as hate speech categories. Similar to the targets of hate speech, the KL divergence between the keyword distributions of the train and test sets is computed. The number of underrepresented keyword categories in the train set is also calculated.

Hate Speech Overlap: In addition to the annotated targets, the overlap between words associated with hate speech can be measured. If different words are more important for the hate label of an example in the test set compared to the the train set, models might have difficulties with generalisation. To this end, a *hate speech score* is calculated as proposed by de Gibert et al. (2018). The hate speech score is based on the Pointwise Mutual Information (PMI), which calculates the correlation of each word w with its respective class $hate$ or $noHate$:

$$HS(w) = PMI(w, hate) - PMI(w, noHate) \tag{5.2}$$

The overlap between the 1500 words with the highest score in the train and test sets is then calculated to examine whether different predictive features for the hate class are used in the train and test sets. For the classification task in the test dataset, which includes the classes *hate*, *offensive* and *normal*, the hate score and the hate score overlap between train and test sets are calculated for the hate and offensive classes and then averaged.

5.2.3 Analysis Results

The results of the analyses described above are presented for the development dataset (Section 5.2.3.1) and the test set (Section 5.2.3.2). Then, Section 5.2.3.3 aims to give some insight into the properties of the clusters.

5.2.3.1 Analysis Results for the Splits on the Development Dataset

The results of the task-agnostic and task-specific analyses are presented in Table 5.2. For the development set, there is a moderate correlation between unigram overlap (Pearson correlation of 0.36 with $p < 0.01$) and the performance of the models: The higher the overlap of unigrams between the train and test set, the higher the model performance. This is not unexpected, although one had actually expected a higher correlation based on the results of Elangovan, He, and Verspoor (2021). They find that for other text

Analysis	Property	Dev. Dataset		Test Dataset	
task-agnostic	Unigram Overlap	0.36	$p < 0.01$	**0.68**	$\boldsymbol{p < 0.01}$
	Sentence Length	−0.06	$p = 0.58$	−0.36	$p = 0.03$
	Rare Words	−0.07	$p = 0.51$	−0.36	$p = 0.03$
task-specific	Hate Speech Overlap	0.13	$p = 0.25$	−0.14	$p = 0.43$
	Split Conversations	0.12	$p = 0.29$	–	
	# Keywords Under-repr.	**0.51**	$\boldsymbol{p < 0.01}$	**0.63**	$\boldsymbol{p < 0.01}$
	KL-Div. Keywords Distr.	0.14	$p = 0.19$	**0.51**	$\boldsymbol{p < 0.01}$
	# Targets Under-repr.	–		0.43	$p = 0.01$
	KL-Div. Targets Distr.	–		0.3	$p = 0.08$
	KL-Div. Data Source Distr.	–		**0.69**	$\boldsymbol{p < 0.01}$

Table 5.2: Pearson correlation between different dataset properties and model performance (F1-score). Important correlations are highlighted in bold. Some analysis methods are dataset-specific and cannot be computed for both datasets.

classification tasks, the unigram overlap between the train and test sets plays a more significant role. For the data splits in this work, however, the moderate correlation suggests that unigram overlap is not the key factor.

Sentence length and rare words do not seem to play a role. No significant correlation can be observed here. Note that the (insignificant) correlations in Table 5.2 are expected to be negative because the longer the sentences and the more rare words there are in the test set, the lower the performance.

The task-specific analysis shows that there is a correlation between the number of underrepresented keyword categories and the dropped performance (Pearson coefficient of 0.51 with $p < 0.01$). These keyword categories, each corresponding to a separate hate keyword as explained above, were used to create the dataset. If some of them are less present in the train set, it might be hard for the model to make the right predictions for them in the test set. Based on this correlation, one might expect the overlap of words with high hate speech scores (i.e. words that are important for the hate class) between the train and test sets to also be high. However, this is not the case, there is no significant correlation because the keywords do not correspond to the words with high hate speech scores. One explanation for this could be that hate keywords that are used for extracting posts, are used not only to extract the hate speech posts, but are also used to extract the whole dataset, so the hate keywords are not necessarily predictive of the hate class.

As explained above, the development dataset consists of different Reddit conversations. Whether these conversations are split between the train and test set is not significant for the performance of the language models.

5.2.3.2 Analysis Results for the Splits on the Test Dataset

On the test dataset, multiple dataset properties correlate with the test-set performance when using the proposed data splits. The scores can be seen in Table 5.2. In contrast to the development dataset, a strong correlation with a coefficient of 0.68, $p < 0.01$ can be observed for unigram overlap. The other task-agnostic analysis methods (sentence length and number of rare words) again show no significant correlations.

For the task-specific analysis, there is no correlation between the overlap of words that are important for the hate (or offensive) class and the test set performance. Given that there is a significant correlation for word overlap, this is somewhat surprising, especially because the (insignificant) correlation is even negative. An explanation might be that the offensive and hate classes include similar words that are predictive of two different classes. For example, the word *bitch* is a predictive feature in the hate example *screw the cowgirls nigger bitches* and in the offensive example *just leave me and my child alone weird ass bitch*.

What seems to be important in the data split, similar to the development dataset, are the keyword categories that are underrepresented in the train set, giving a correlation of 0.63, $p < 0.01$.

However, another positive, significant correlation exists: The performance decrease correlates with the distribution shift (measured by the KL divergence) of different data sources between the train and test set (Pearson coefficient of 0.69 with $p < 0.001$). As described in Section 3.1.2, the test dataset consists of two data sources, Gab (46%) and Twitter (54%). Closest-split leads to an imbalance of the two data sources in the train and test sets, which correlates with decreased performance. For example in the HateBert-bottleneck-50-closest-split, 75% of the test set consists of Gab examples. Which data source dominates the test set differs in different closest-splits.

Set	Class	Topics
Test	hate	nigger, moslem, white, black, jews
	offensive	nigger, moslem, white, jews, kike
	normal	moslem, nigger, white, black, people
Train	hate	nigger, moslem, white, black, jews
	offensive	bitch, white, ghetto, woman, faggot
	normal	white, people, women, raped, violence

Table 5.3: Top 5 topics for different classes in the test dataset. The topics are obtained from train and test sets of the closest-split with latent representations from HateBert.

5.2.3.3 Properties of the Clusters that Form the Closest-Split

Having looked at the properties of the train and test split, the following paragraphs take a closer look at the clusters, that form the basis of the proposed splitting approaches.

For closest-split, usually between 10-15 clusters are sufficient to create the split, even though up to 50 clusters are allowed in the algorithm (for the subset-sum split it is usually between 20-30). One might expect that when clustering the hidden representations, the clusters would correspond to one class. This is the case in the binary development dataset, where one class usually dominates a cluster. For the test dataset, however, often a similar amount of hate and offensive examples are present in the clusters, though the number of normal examples again differs.

To get an intuition for the kind of topics that are included in the clusters, topic clusters are extracted using c-TF-IDF, a modified version of TF-IDF as proposed by Grootendorst (2022). As an example, Table 5.3 summarises the topics with the highest c-TF-IDF scores of the clusters in the train and test sets for each class of a HateBert-bottleneck-50 model. Looking at the topics of the clusters in the train and test set, there seems to be a tendency for offensive and normal classes to have different topics in the train and test sets, while the hate class has similar ones. A manual analysis of cluster topics for all cluster splits did not lead to conclusive results. Many of the topics found by c-TF-IDF seem to coincide with the targets that were annotated, and used for the analysis in the previous section. No strong correlation between targets and performance was observed then, which strengthens the result that different targets in the train and test sets are not the reason for the decreased performance.

Chapter 6

Conclusion

6.1 Conclusion

O.o.d. generalisation is important for language models because the data distribution a model encounters at deployment time often differs from the data distribution seen during training. Generalising successfully is especially important for socially-relevant tasks like hate speech detection.

This work presents a new data-splitting method, designed to test the robustness of models more rigorously than the often-used random split. Hate speech is used as the testbed for developing and evaluating the new splitting method. The new data split, closest-split, is based on models' internal representations, rather than meta-data annotations or input text properties. This makes the split task- and dataset-specific, without relying on costly annotations.

The results in Chapter 4 reveal dramatic drops in test set performance in comparison to a random split. Closest-split leads to performance drops in a range of different scenarios in the domain of hate speech detection: for different models and datasets, for binary and multi-class classification, for a task that models can solve almost perfectly on a random split and for a task with weaker model performance. Moreover, while each data split is obtained using the internal representations of a specific model, the splits are similarly challenging for other models. This means the method can be used to create a challenging split for use across models.

The analyses in Chapter 5 show that the properties of the train and test sets differ from dataset to dataset. Since there is no property that clearly correlates with decreased model performance for both datasets, closest-split cannot be easily replicated based on these properties, and the latent representations need to be used. This makes closest-split an interesting alternative to existing data-splitting approaches, uncovering weaknesses in models that are otherwise not easily identified.

6.2 Limitations and Future Work

This work uses hate speech detection as the test bed for developing and evaluating a data-splitting method, but the approach could be applied much more widely to improve generalisation evaluation for other NLP tasks. Although closest-split is designed to be task-agnostic, future work should confirm wider applicability. Similarly, the work has not explored topic-specific datasets. Due to the lower variance across examples, closest-split might not be as effective for these.

The robustness of the method to different settings seems convincing, producing challenging splits for different language models, different hidden representation sizes, between models, etc. However, this work has not explicitly tested the sensitivity of the method to other hyperparameters. For example, the size of the test set could have an impact. Since the data splits are based on the internal representations of a model, model hyperparameters, such as longer training, different regularisation or different scheduling, could change the hidden representations. This in turn could also lead to different data splits and different final results.

With regards to the analyses in Chapter 5, although a wide range of data properties was considered, the possibility that other factors can explain the decreased performance cannot be ruled out. It is possible that other meta-properties, e.g., the authorship of the posts, correlate with decreased performance. But further properties are not annotated in the considered datasets and could therefore not be analysed here.

Future work could investigate whether known measures to improve generalisation also make models more robust to closest-split. Koufakou et al. (2020), for example, use features based on a hate lexicon to enhance the embeddings of a BERT model and achieve better results in cross-domain generalisation. This procedure might help to classify examples whose keyword categories are under-represented in the closest-split train set. Another challenging factor in the data split was the shift in data sources between the train and test sets. Merging different hate speech datasets might address this issue and improve generalisation (Salminen et al., 2020; Chiril et al., 2022; Bourgeade et al., 2023). It would be interesting to see whether new insights could be gained into these generalisation methods using closest-split.

References

Aggarwal, Charu C., Hinneburg, Alexander, and Keim, Daniel A., 2001. On the surprising behavior of distance metrics in high dimensional space. In: Jan Van den Bussche and Victor Vianu, eds. *Database theory — icdt 2001*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp.420–434.

Anzovino, Maria, Fersini, Elisabetta, and Rosso, Paolo, 2018. Automatic identification and classification of misogynistic language on twitter [Online]. In: pp.57–64. Available from: https://doi.org/10.1007/978-3-319-91947-8_6.

Arango, Aymé, Pérez, Jorge, and Poblete, Barbara, 2019. Hate speech detection is not as easy as you may think: a closer look at model validation. *Proceedings of the 42nd international acm sigir conference on research and development in information retrieval* [Online], Sigir'19. Paris, France: Association for Computing Machinery, pp.45–54. Available from: https://doi.org/10.1145/3331184.3331262.

Arora, Udit, Huang, William, and He, He, 2021. Types of out-of-distribution texts and how to detect them. *Proceedings of the 2021 conference on empirical methods in natural language processing* [Online]. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, pp.10687–10701. Available from: https://doi.org/10.18653/v1/2021.emnlp-main.835.

Basile, Valerio, Bosco, Cristina, Fersini, Elisabetta, Nozza, Debora, Patti, Viviana, Rangel Pardo, Francisco Manuel, Rosso, Paolo, and Sanguinetti, Manuela, 2019. SemEval-2019 task 5: multilingual detection of hate speech against immigrants and women in Twitter. *Proceedings of the 13th international workshop on semantic evaluation* [Online]. Minneapolis, Minnesota, USA: Association for Computational Linguistics, pp.54–63. Available from: https://doi.org/10.18653/v1/S19-2007.

Beyer, Kevin, Goldstein, Jonathan, Ramakrishnan, Raghu, and Shaft, Uri, 1999. When is “nearest neighbor” meaningful? In: Catriel Beeri and Peter Buneman, eds. *Database theory — icdt’99*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp.217–235.

Bhargava, Prajjwal, Drozd, Aleksandr, and Rogers, Anna, 2021. *Generalization in nli: ways (not) to go beyond simple heuristics*. arXiv: 2110.01518 [cs.CL].

Bourgeade, Tom, Chiril, Patricia, Benamara, Farah, and Moriceau, Véronique, 2023. What did you learn to hate? a topic-oriented analysis of generalization in hate speech detection. *Proceedings of the 17th conference of the european chapter of the association for computational linguistics* [Online]. Dubrovnik, Croatia: Association for Computational Linguistics, pp.3495–3508. Available from: https://aclanthology.org/2023.eacl-main.254.

Broscheit, Samuel, Do, Quynh, and Gaspers, Judith, 2022. Distributionally robust finetuning BERT for covariate drift in spoken language understanding. *Proceedings of the 60th annual meeting of the association for computational linguistics (volume 1: long papers)* [Online]. Dublin, Ireland: Association for Computational Linguistics, pp.1970–1985. Available from: https://doi.org/10.18653/v1/2022.acl-long.139.

Brysbaert, Marc and New, Boris, 2009. Moving beyond kucera and francis: a critical evaluation of current word frequency norms and the introduction of a new and improved word frequency measure for american english. *Behavior research methods* [Online], 41, pp.977–90. Available from: https://doi.org/10.3758/BRM.41.4.977.

Caselli, Tommaso, Basile, Valerio, Mitrović, Jelena, and Granitzer, Michael, 2021. HateBERT: retraining BERT for abusive language detection in English. *Proceedings of the 5th workshop on online abuse and harms (woah 2021)* [Online]. Online: Association for Computational Linguistics, pp.17–25. Available from: https://doi.org/10.18653/v1/2021.woah-1.3.

Caselli, Tommaso, Basile, Valerio, Mitrović, Jelena, Kartoziya, Inga, and Granitzer, Michael, 2020. I feel offended, don’t be abusive! implicit/explicit messages in offensive and abusive language. *Proceedings of the twelfth language resources and evaluation conference* [Online]. Marseille, France: European Language Resources Association, pp.6193–6202. Available from: https://aclanthology.org/2020.lrec-1.760.

Chen, Michael, D'Arcy, Mike, Liu, Alisa, Fernandez, Jared, and Downey, Doug, 2019. CODAH: an adversarially-authored question answering dataset for common sense. *Proceedings of the 3rd workshop on evaluating vector space representations for NLP* [Online]. Minneapolis, USA: Association for Computational Linguistics, pp.63–69. Available from: https://doi.org/10.18653/v1/W19-2008.

Chiril, Patricia, Pamungkas, Endang Wahyu, Benamara, Farah, Moriceau, Véronique, and Patti, Viviana, 2022. Emotionally informed hate speech detection: a multi-target perspective. *Cognitive computation* [Online], 14(1), pp.322–352. Available from: https://doi.org/10.1007/s12559-021-09862-5.

Cho, Kyunghyun, Merriënboer, Bart van, Gulcehre, Caglar, Bahdanau, Dzmitry, Bougares, Fethi, Schwenk, Holger, and Bengio, Yoshua, 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* [Online]. Doha, Qatar: Association for Computational Linguistics, pp.1724–1734. Available from: https://doi.org/10.3115/v1/D14-1179.

Chowdhery, Aakanksha, Narang, Sharan, Devlin, Jacob, Bosma, Maarten, Mishra, Gaurav, Roberts, Adam, Barham, Paul, Chung, Hyung Won, Sutton, Charles, Gehrmann, Sebastian, Schuh, Parker, Shi, Kensen, Tsvyashchenko, Sasha, Maynez, Joshua, Rao, Abhishek, Barnes, Parker, Tay, Yi, Shazeer, Noam, Prabhakaran, Vinodkumar, Reif, Emily, Du, Nan, Hutchinson, Ben, Pope, Reiner, Bradbury, James, Austin, Jacob, Isard, Michael, Gur-Ari, Guy, Yin, Pengcheng, Duke, Toju, Levskaya, Anselm, Ghemawat, Sanjay, Dev, Sunipa, Michalewski, Henryk, Garcia, Xavier, Misra, Vedant, Robinson, Kevin, Fedus, Liam, Zhou, Denny, Ippolito, Daphne, Luan, David, Lim, Hyeontaek, Zoph, Barret, Spiridonov, Alexander, Sepassi, Ryan, Dohan, David, Agrawal, Shivani, Omernick, Mark, Dai, Andrew M., Pillai, Thanumalayan Sankaranarayana, Pellat, Marie, Lewkowycz, Aitor, Moreira, Erica, Child, Rewon, Polozov, Oleksandr, Lee, Katherine, Zhou, Zongwei, Wang, Xuezhi, Saeta, Brennan, Diaz, Mark, Firat, Orhan, Catasta, Michele, Wei, Jason, Meier-Hellstern, Kathy, Eck, Douglas, Dean, Jeff, Petrov, Slav, and Fiedel, Noah, 2022. *Palm: scaling language modeling with pathways*. arXiv: 2204.02311 [cs.CL].

Davidson, Thomas, Warmsley, Dana, Macy, Michael, and Weber, Ingmar, 2017. Automated hate speech detection and the problem of offensive language. *Proceedings of the* *international aaai conference on web and social media* [Online], 11(1), pp.512–515. Available from: https://doi.org/10.1609/icwsm.v11i1.14955.

de Gibert, Ona, Pérez, Naiara, Pablos, Aitor García, and Cuadros, Montse, 2018. Hate speech dataset from a white supremacy forum. *Corr* [Online], abs/1809.04444. arXiv: 1809.04444. Available from: http://arxiv.org/abs/1809.04444.

Devlin, Jacob, Chang, Ming-Wei, Lee, Kenton, and Toutanova, Kristina, 2019. BERT: pre-training of deep bidirectional transformers for language understanding. *Proceedings of the 2019 conference of the north American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)* [Online]. Minneapolis, Minnesota: Association for Computational Linguistics, pp.4171–4186. Available from: https://doi.org/10.18653/v1/N19-1423.

Dixon, Lucas, Li, John, Sorensen, Jeffrey, Thain, Nithum, and Vasserman, Lucy, 2018. Measuring and mitigating unintended bias in text classification. *Proceedings of the 2018 aaai/acm conference on ai, ethics, and society* [Online], Aies '18. New Orleans, LA, USA: Association for Computing Machinery, pp.67–73. Available from: https://doi.org/10.1145/3278721.3278729.

Ebrahimi, Javid, Rao, Anyi, Lowd, Daniel, and Dou, Dejing, 2018. HotFlip: white-box adversarial examples for text classification. *Proceedings of the 56th annual meeting of the association for computational linguistics (volume 2: short papers)* [Online]. Melbourne, Australia: Association for Computational Linguistics, pp.31–36. Available from: https://doi.org/10.18653/v1/P18-2006.

Elangovan, Aparna, He, Jiayuan, and Verspoor, Karin, 2021. Memorization vs. generalization : quantifying data leakage in NLP performance evaluation. *Proceedings of the 16th conference of the european chapter of the association for computational linguistics: main volume* [Online]. Online: Association for Computational Linguistics, pp.1325–1335. Available from: https://doi.org/10.18653/v1/2021.eacl-main.113.

ElSherief, Mai, Nilizadeh, Shirin, Nguyen, Dana, Vigna, Giovanni, and Belding, Elizabeth, 2018. Peer to peer hate: hate speech instigators and their targets. *Proceedings of the international aaai conference on web and social media* [Online], 12(1). Available from: https://doi.org/10.1609/icwsm.v12i1.15038.

ElSherief, Mai, Ziems, Caleb, Muchlinski, David, Anupindi, Vaishnavi, Seybolt, Jordyn, De Choudhury, Munmun, and Yang, Diyi, 2021. Latent hatred: a benchmark for understanding implicit hate speech. *Proceedings of the 2021 conference on empirical methods in natural language processing* [Online]. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, pp.345–363. Available from: https://doi.org/10.18653/v1/2021.emnlp-main.29.

Feng, Steven Y., Gangal, Varun, Wei, Jason, Chandar, Sarath, Vosoughi, Soroush, Mitamura, Teruko, and Hovy, Eduard, 2021. A survey of data augmentation approaches for NLP. *Findings of the association for computational linguistics: acl-ijcnlp 2021* [Online]. Online: Association for Computational Linguistics, pp.968–988. Available from: https://doi.org/10.18653/v1/2021.findings-acl.84.

Fersini, Elisabetta, Nozza, Debora, and Rosso, Paolo, 2018. Overview of the evalita 2018 task on automatic misogyny identification (ami). *Evalita@clic-it*.

Fiske, Susan, Cuddy, Amy, Glick, Peter, and Xu, J., 2002. A model of (often mixed) stereotype content: competence and warmth respectively follow from perceived status and competition. *Journal of personality and social psychology* [Online], 82, pp.878–902. Available from: https://doi.org/10.1037/0022-3514.82.6.878.

Fortuna, Paula, Bonavita, Ilaria, and Nunes, Sérgio, 2018. Merging datasets for hate speech classification in italian [Online]. In: pp.218–223. Available from: https://doi.org/10.4000/books.aaccademia.4752.

Fortuna, Paula and Nunes, Sérgio, 2018. A survey on automatic detection of hate speech in text. *Acm comput. surv.* [Online], 51(4). Available from: https://doi.org/10.1145/3232676.

Founta, Antigoni, Djouvas, Constantinos, Chatzakou, Despoina, Leontiadis, Ilias, Blackburn, Jeremy, Stringhini, Gianluca, Vakali, Athena, Sirivianos, Michael, and Kourtellis, Nicolas, 2018. Large scale crowdsourcing and characterization of twitter abusive behavior. *Proceedings of the international aaai conference on web and social media* [Online], 12(1). Available from: https://doi.org/10.1609/icwsm.v12i1.14991.

Fraser, Kathleen C., Nejadgholi, Isar, and Kiritchenko, Svetlana, 2021. Understanding and countering stereotypes: a computational approach to the stereotype content model. *Proceedings of the 59th annual meeting of the association for computational linguistics and the 11th international joint conference on natural language processing (volume 1:*

long papers) [Online]. Online: Association for Computational Linguistics, pp.600–616. Available from: https://doi.org/10.18653/v1/2021.acl-long.50.

Gardner, Matt, Artzi, Yoav, Basmov, Victoria, Berant, Jonathan, Bogin, Ben, Chen, Sihao, Dasigi, Pradeep, Dua, Dheeru, Elazar, Yanai, Gottumukkala, Ananth, Gupta, Nitish, Hajishirzi, Hannaneh, Ilharco, Gabriel, Khashabi, Daniel, Lin, Kevin, Liu, Jiangming, Liu, Nelson F., Mulcaire, Phoebe, Ning, Qiang, Singh, Sameer, Smith, Noah A., Subramanian, Sanjay, Tsarfaty, Reut, Wallace, Eric, Zhang, Ally, and Zhou, Ben, 2020. Evaluating models' local decision boundaries via contrast sets. *Findings of the association for computational linguistics: emnlp 2020* [Online]. Online: Association for Computational Linguistics, pp.1307–1323. Available from: https://doi.org/10.18653/v1/2020.findings-emnlp.117.

Godbole, Ameya and Jia, Robin, 2022. *Benchmarking long-tail generalization with likelihood splits*. arXiv: 2210.06799 [cs.CL].

Gorman, Kyle and Bedrick, Steven, 2019. We need to talk about standard splits. *Proceedings of the 57th annual meeting of the association for computational linguistics* [Online]. Florence, Italy: Association for Computational Linguistics, pp.2786–2791. Available from: https://doi.org/10.18653/v1/P19-1267.

Grootendorst, Maarten, 2022. *Bertopic: neural topic modeling with a class-based tf-idf procedure*. arXiv: 2203.05794 [cs.CL].

Gururangan, Suchin, Swayamdipta, Swabha, Levy, Omer, Schwartz, Roy, Bowman, Samuel, and Smith, Noah A., 2018. Annotation artifacts in natural language inference data. *Proceedings of the 2018 conference of the north American chapter of the association for computational linguistics: human language technologies, volume 2 (short papers)* [Online]. New Orleans, Louisiana: Association for Computational Linguistics, pp.107–112. Available from: https://doi.org/10.18653/v1/N18-2017.

Huang, Xiaolei and Paul, Michael J., 2019. Neural user factor adaptation for text classification: learning to generalize across author demographics. *Proceedings of the eighth joint conference on lexical and computational semantics (*SEM 2019)* [Online]. Minneapolis, Minnesota: Association for Computational Linguistics, pp.136–146. Available from: https://doi.org/10.18653/v1/S19-1015.

Hupkes, Dieuwke, Giulianelli, Mario, Dankers, Verna, Artetxe, Mikel, Elazar, Yanai, Pimentel, Tiago, Christodoulopoulos, Christos, Lasri, Karim, Saphra, Naomi, Sinclair, Arabella, Ulmer, Dennis, Schottmann, Florian, Batsuren, Khuyagbaatar, Sun, Kaiser, Sinha, Koustuv, Khalatbari, Leila, Ryskina, Maria, Frieske, Rita, Cotterell, Ryan, and Jin, Zhijing, 2023. *State-of-the-art generalisation research in nlp: a taxonomy and review*. arXiv: 2210.03050 [cs.CL].

Karan, Mladen and Šnajder, Jan, 2018. Cross-domain detection of abusive language online. *Proceedings of the 2nd workshop on abusive language online (ALW2)* [Online]. Brussels, Belgium: Association for Computational Linguistics, pp.132–137. Available from: https://doi.org/10.18653/v1/W18-5117.

Kaushik, Divyansh, Hovy, Eduard H., and Lipton, Zachary C., 2019. Learning the difference that makes a difference with counterfactually-augmented data. *Corr* [Online], abs/1909.12434. arXiv: 1909.12434. Available from: http://arxiv.org/abs/1909.12434.

Kellerer, Hans, Pferschy, Ulrich, and Pisinger, David, 2004. The subset sum problem. In: *Knapsack problems* [Online]. Berlin, Heidelberg: Springer Berlin Heidelberg, pp.73–115. Available from: https://doi.org/10.1007/978-3-540-24777-7_4.

Kennedy, Brendan, Jin, Xisen, Mostafazadeh Davani, Aida, Dehghani, Morteza, and Ren, Xiang, 2020. Contextualizing hate speech classifiers with post-hoc explanation. *Proceedings of the 58th annual meeting of the association for computational linguistics* [Online]. Online: Association for Computational Linguistics, pp.5435–5442. Available from: https://doi.org/10.18653/v1/2020.acl-main.483.

Keysers, Daniel, Schärli, Nathanael, Scales, Nathan, Buisman, Hylke, Furrer, Daniel, Kashubin, Sergii, Momchev, Nikola, Sinopalnikov, Danila, Stafiniak, Lukasz, Tihon, Tibor, Tsarkov, Dmitry, Wang, Xiao, Zee, Marc van, and Bousquet, Olivier, 2019. Measuring compositional generalization: A comprehensive method on realistic data. *Corr* [Online], abs/1912.09713. arXiv: 1912.09713. Available from: http://arxiv.org/abs/1912.09713.

Kiela, Douwe, Bartolo, Max, Nie, Yixin, Kaushik, Divyansh, Geiger, Atticus, Wu, Zhengxuan, Vidgen, Bertie, Prasad, Grusha, Singh, Amanpreet, Ringshia, Pratik, Ma, Zhiyi, Thrush, Tristan, Riedel, Sebastian, Waseem, Zeerak, Stenetorp, Pontus, Jia, Robin, Bansal, Mohit, Potts, Christopher, and Williams, Adina, 2021. Dynabench: rethinking benchmarking in NLP. *Proceedings of the 2021 conference of the north american chapter of the association for computational linguistics: human language technologies*

[Online]. Online: Association for Computational Linguistics, pp.4110–4124. Available from: https://doi.org/10.18653/v1/2021.naacl-main.324.

Kim, Najoung and Linzen, Tal, 2020. COGS: a compositional generalization challenge based on semantic interpretation. *Proceedings of the 2020 conference on empirical methods in natural language processing (emnlp)* [Online]. Online: Association for Computational Linguistics, pp.9087–9105. Available from: https://doi.org/10.18653/v1/2020.emnlp-main.731.

Koufakou, Anna, Pamungkas, Endang Wahyu, Basile, Valerio, and Patti, Viviana, 2020. HurtBERT: incorporating lexical features with BERT for the detection of abusive language. *Proceedings of the fourth workshop on online abuse and harms* [Online]. Online: Association for Computational Linguistics, pp.34–43. Available from: https://doi.org/10.18653/v1/2020.alw-1.5.

Kullback, Solomon and Leibler, Richard A, 1951. On information and sufficiency. *The annals of mathematical statistics*, 22(1), pp.79–86.

Kurrek, Jana, Saleem, Haji Mohammad, and Ruths, Derek, 2020. Towards a comprehensive taxonomy and large-scale annotated corpus for online slur usage. *Proceedings of the fourth workshop on online abuse and harms* [Online]. Online: Association for Computational Linguistics, pp.138–149. Available from: https://doi.org/10.18653/v1/2020.alw-1.17.

Lake, Brenden M. and Baroni, Marco, 2017. Still not systematic after all these years: on the compositional skills of sequence-to-sequence recurrent networks. *Corr* [Online], abs/1711.00350. arXiv: 1711.00350. Available from: http://arxiv.org/abs/1711.00350.

Lazaridou, Angeliki, Kuncoro, Adhi, Gribovskaya, Elena, Agrawal, Devang, Liska, Adam, Terzi, Tayfun, Gimenez, Mai, Masson d'Autume, Cyprien de, Kocisky, Tomas, Ruder, Sebastian, Yogatama, Dani, Cao, Kris, Young, Susannah, and Blunsom, Phil, 2021. Mind the gap: assessing temporal generalization in neural language models. In: M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, eds. *Advances in neural information processing systems* [Online]. Vol. 34. Curran Associates, Inc., pp.29348–29363. Available from: https://proceedings.neurips.cc/paper_files/paper/2021/file/f5bf0ba0a17ef18f9607774722f5698c-Paper.pdf.

Liu, Ping, Li, Wen, and Zou, Liang, 2019. NULI at SemEval-2019 task 6: transfer learning for offensive language detection using bidirectional transformers. *Proceedings of the 13th international workshop on semantic evaluation* [Online]. Minneapolis, Minnesota, USA: Association for Computational Linguistics, pp.87–91. Available from: https://doi.org/10.18653/v1/S19-2011.

Liu, Yinhan, Ott, Myle, Goyal, Naman, Du, Jingfei, Joshi, Mandar, Chen, Danqi, Levy, Omer, Lewis, Mike, Zettlemoyer, Luke, and Stoyanov, Veselin, 2019. Roberta: A robustly optimized BERT pretraining approach. *Corr* [Online], abs/1907.11692. arXiv: 1907.11692. Available from: http://arxiv.org/abs/1907.11692.

Lloyd, S., 1982. Least squares quantization in pcm. *Ieee transactions on information theory* [Online], 28(2), pp.129–137. Available from: https://doi.org/10.1109/TIT.1982.1056489.

Malinin, Andrey, Band, Neil, Gal, Yarin, Gales, Mark, Ganshin, Alexander, Chesnokov, German, Noskov, Alexey, Ploskonosov, Andrey, Prokhorenkova, Liudmila, Provilkov, Ivan, Raina, Vatsal, Raina, Vyas, Roginskiy, Denis, Shmatova, Mariya, Tigas, Panagiotis, and Yangel, Boris, 2021. Shifts: a dataset of real distributional shift across multiple large-scale tasks. In: J. Vanschoren and S. Yeung, eds. *Proceedings of the neural information processing systems track on datasets and benchmarks* [Online]. Vol. 1. Curran. Available from: https://datasets-benchmarks-proceedings.neurips.cc/paper_files/paper/2021/file/ad61ab143223efbc24c7d2583be69251-Paper-round2.pdf.

Mathew, Binny, Dutt, Ritam, Goyal, Pawan, and Mukherjee, Animesh, 2019. Spread of hate speech in online social media. *Proceedings of the 10th acm conference on web science* [Online], Websci '19. Boston, Massachusetts, USA: Association for Computing Machinery, pp.173–182. Available from: https://doi.org/10.1145/3292522.3326034.

Mathew, Binny, Saha, Punyajoy, Yimam, Seid Muhie, Biemann, Chris, Goyal, Pawan, and Mukherjee, Animesh, 2020. Hatexplain: A benchmark dataset for explainable hate speech detection. *Corr* [Online], abs/2012.10289. arXiv: 2012.10289. Available from: https://arxiv.org/abs/2012.10289.

May, Chandler, Wang, Alex, Bordia, Shikha, Bowman, Samuel R., and Rudinger, Rachel, 2019. On measuring social biases in sentence encoders. *Proceedings of the 2019 conference of the north American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)* [Online]. Minneapolis, Minnesota: Association for Computational Linguistics, pp.622–628. Available from: https://doi.org/10.18653/v1/N19-1063.

McCoy, Tom, Pavlick, Ellie, and Linzen, Tal, 2019. Right for the wrong reasons: diagnosing syntactic heuristics in natural language inference. *Proceedings of the 57th annual meeting of the association for computational linguistics* [Online]. Florence, Italy: Association for Computational Linguistics, pp.3428–3448. Available from: https://doi.org/10.18653/v1/P19-1334.

McInnes, Leland, Healy, John, and Melville, James, 2020. *Umap: uniform manifold approximation and projection for dimension reduction*. arXiv: 1802.03426 [stat.ML].

Michel, Paul and Neubig, Graham, 2018. MTNT: a testbed for machine translation of noisy text. *Proceedings of the 2018 conference on empirical methods in natural language processing* [Online]. Brussels, Belgium: Association for Computational Linguistics, pp.543–553. Available from: https://doi.org/10.18653/v1/D18-1050.

Miller, Timothy, Laparra, Egoitz, and Bethard, Steven, 2021. Domain adaptation in practice: lessons from a real-world information extraction pipeline. *Proceedings of the second workshop on domain adaptation for nlp* [Online]. Kyiv, Ukraine: Association for Computational Linguistics, pp.105–110. Available from: https://aclanthology.org/2021.adaptnlp-1.11.

Naik, Aakanksha, Lehman, Jill, and Rosé, Carolyn, 2022. Adapting to the long tail: a meta-analysis of transfer learning research for language understanding tasks. *Transactions of the association for computational linguistics* [Online], 10, pp.956–980. Available from: https://doi.org/10.1162/tacl_a_00500.

Naik, Aakanksha, Ravichander, Abhilasha, Sadeh, Norman, Rose, Carolyn, and Neubig, Graham, 2018. Stress test evaluation for natural language inference. *Proceedings of the 27th international conference on computational linguistics* [Online]. Santa Fe, New Mexico, USA: Association for Computational Linguistics, pp.2340–2353. Available from: https://aclanthology.org/C18-1198.

Nejadgholi, Isar and Kiritchenko, Svetlana, 2020. On cross-dataset generalization in automatic detection of online abuse. *Proceedings of the fourth workshop on online abuse and harms* [Online]. Online: Association for Computational Linguistics, pp.173–183. Available from: https://doi.org/10.18653/v1/2020.alw-1.20.

Nishida, Noriki and Matsumoto, Yuji, 2022. Out-of-domain discourse dependency parsing via bootstrapping: an empirical analysis on its effectiveness and limitation. *Transactions of the association for computational linguistics* [Online], 10, pp.127–144. Available from: https://doi.org/10.1162/tacl_a_00451.

Ousidhoum, Nedjma, Lin, Zizheng, Zhang, Hongming, Song, Yangqiu, and Yeung, Dit-Yan, 2019. Multilingual and multi-aspect hate speech analysis. *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (emnlp-ijcnlp)* [Online]. Hong Kong, China: Association for Computational Linguistics, pp.4675–4684. Available from: https://doi.org/10.18653/v1/D19-1474.

Palmer, Alexis, Carr, Christine, Robinson, Melissa, and Sanders, Jordan, 2020. Cold: annotation scheme and evaluation data set for complex offensive language in english. *Journal for language technology and computational linguistics* [Online], 34(1), pp.1–28. Available from: https://doi.org/10.21248/jlcl.34.2020.222.

Pyysalo, Sampo, Kanerva, Jenna, Virtanen, Antti, and Ginter, Filip, 2021. WikiBERT models: deep transfer learning for many languages. *Proceedings of the 23rd nordic conference on computational linguistics (nodalida)* [Online]. Reykjavik, Iceland (Online): Linköping University Electronic Press, Sweden, pp.1–10. Available from: https://aclanthology.org/2021.nodalida-main.1.

Qian, Jing, Bethke, Anna, Liu, Yinyin, Belding, Elizabeth, and Wang, William Yang, 2019. A benchmark dataset for learning to intervene in online hate speech. *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (emnlp-ijcnlp)* [Online]. Hong Kong, China: Association for Computational Linguistics, pp.4755–4764. Available from: https://doi.org/10.18653/v1/D19-1482.

Qiao, Yifan, Xiong, Chenyan, Liu, Zhenghao, and Liu, Zhiyuan, 2019. Understanding the behaviors of BERT in ranking. *Corr* [Online], abs/1904.07531. arXiv: 1904.07531. Available from: http://arxiv.org/abs/1904.07531.

Rabanser, Stephan, Günnemann, Stephan, and Lipton, Zachary, 2019. Failing loudly: an empirical study of methods for detecting dataset shift. In: H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds. *Advances in neural information processing systems* [Online]. Vol. 32. Curran Associates, Inc. Available from: https://proceedings.neurips.cc/paper_files/paper/2019/file/846c260d715e5b854ffad5f70a516c88-Paper.pdf.

Ramesh Kashyap, Abhinav, Mehnaz, Laiba, Malik, Bhavitvya, Waheed, Abdul, Hazarika, Devamanyu, Kan, Min-Yen, and Shah, Rajiv Ratn, 2021. Analyzing the domain robustness of pretrained language models, layer by layer. *Proceedings of the second workshop on domain adaptation for nlp* [Online]. Kyiv, Ukraine: Association for Computational Linguistics, pp.222–244. Available from: https://aclanthology.org/2021.adaptnlp-1.23.

Razeghi, Yasaman, Logan IV, Robert L, Gardner, Matt, and Singh, Sameer, 2022. Impact of pretraining term frequencies on few-shot numerical reasoning. *Findings of the association for computational linguistics: emnlp 2022* [Online]. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, pp.840–854. Available from: https://aclanthology.org/2022.findings-emnlp.59.

Reimers, Nils and Gurevych, Iryna, 2019. Sentence-BERT: sentence embeddings using Siamese BERT-networks. *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (emnlp-ijcnlp)* [Online]. Hong Kong, China: Association for Computational Linguistics, pp.3982–3992. Available from: https://doi.org/10.18653/v1/D19-1410.

Roth, Karsten, Pemula, Latha, Zepeda, Joaquin, Schölkopf, Bernhard, Brox, Thomas, and Gehler, Peter, 2022. Towards total recall in industrial anomaly detection. *2022 ieee/cvf conference on computer vision and pattern recognition (cvpr)* [Online], pp.14298–14308. Available from: https://doi.org/10.1109/CVPR52688.2022.01392.

Salminen, Joni, Hopf, Maximilian, Chowdhury, Shammur A., Jung, Soon-gyo, Almerekhi, Hind, and Jansen, Bernard J., 2020. Developing an online hate classifier for multiple social media platforms. *Human-centric computing and information sciences* [Online], 10(1), p.1. Available from: https://doi.org/10.1186/s13673-019-0205-6.

Sia, Suzanna, Dalmia, Ayush, and Mielke, Sabrina J., 2020. Tired of topic models? clusters of pretrained word embeddings make for fast and good topics too! *Proceedings of the 2020 conference on empirical methods in natural language processing (emnlp)* [Online]. Online: Association for Computational Linguistics, pp.1728–1736. Available from: https://doi.org/10.18653/v1/2020.emnlp-main.135.

Sinha, Koustuv, Jia, Robin, Hupkes, Dieuwke, Pineau, Joelle, Williams, Adina, and Kiela, Douwe, 2021. Masked language modeling and the distributional hypothesis: order word matters pre-training for little. *Proceedings of the 2021 conference on empirical methods in natural language processing* [Online]. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, pp.2888–2913. Available from: https://doi.org/10.18653/v1/2021.emnlp-main.230.

Søgaard, Anders, 2020. Some languages seem easier to parse because their treebanks leak. *Proceedings of the 2020 conference on empirical methods in natural language processing (emnlp)* [Online]. Online: Association for Computational Linguistics, pp.2765–2770. Available from: https://doi.org/10.18653/v1/2020.emnlp-main.220.

Søgaard, Anders, Ebert, Sebastian, Bastings, Jasmijn, and Filippova, Katja, 2021. We need to talk about random splits. *Proceedings of the 16th conference of the european chapter of the association for computational linguistics: main volume* [Online]. Online: Association for Computational Linguistics, pp.1823–1832. Available from: https://doi.org/10.18653/v1/2021.eacl-main.156.

Speer, Robyn, 2022. *Rspeer/wordfreq: v3.0* (v.v3.0.2) [Online]. Zenodo. Available from: https://doi.org/10.5281/zenodo.7199437.

Swamy, Steve Durairaj, Jamatia, Anupam, and Gambäck, Björn, 2019. Studying generalisability across abusive language detection datasets. *Proceedings of the 23rd conference on computational natural language learning (conll)* [Online]. Hong Kong, China: Association for Computational Linguistics, pp.940–950. Available from: https://doi.org/10.18653/v1/K19-1088.

Talat, Zeerak, Thorne, James, and Bingel, Joachim, 2018. Bridging the gaps: multi task learning for domain transfer of hate speech detection. In: *Online harassment* [Online]. Ed. by Jennifer Golbeck. Cham: Springer International Publishing, pp.29–55. Available from: https://doi.org/10.1007/978-3-319-78583-7_3.

Thompson, Laure and Mimno, David, 2020. Topic modeling with contextualized word representation clusters. *Corr* [Online], abs/2010.12626. arXiv: 2010.12626. Available from: https://arxiv.org/abs/2010.12626.

Timkey, William and Schijndel, Marten van, 2021. All bark and no bite: rogue dimensions in transformer language models obscure representational quality. *Proceedings of the 2021 conference on empirical methods in natural language processing* [Online]. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, pp.4527–4546. Available from: https://doi.org/10.18653/v1/2021.emnlp-main.372.

Turc, Iulia, Chang, Ming-Wei, Lee, Kenton, and Toutanova, Kristina, 2019. Well-read students learn better: the impact of student initialization on knowledge distillation. *Corr* [Online], abs/1908.08962. arXiv: 1908.08962. Available from: http://arxiv.org/abs/1908.08962.

Varis, Dusan and Bojar, Ondřej, 2021. Sequence length is a domain: length-based overfitting in transformer models. *Proceedings of the 2021 conference on empirical methods in natural language processing* [Online]. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, pp.8246–8257. Available from: https://doi.org/10.18653/v1/2021.emnlp-main.650.

Vaswani, Ashish, Shazeer, Noam, Parmar, Niki, Uszkoreit, Jakob, Jones, Llion, Gomez, Aidan N, Kaiser, Lukasz, and Polosukhin, Illia, 2017. Attention is all you need. In: I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds. *Advances in neural information processing systems* [Online]. Vol. 30. Curran Associates, Inc. Available from: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Vidgen, Bertie, Harris, Alex, Nguyen, Dong, Tromble, Rebekah, Hale, Scott, and Margetts, Helen, 2019. Challenges and frontiers in abusive content detection. *Proceedings of the third workshop on abusive language online* [Online]. Florence, Italy: Association for Computational Linguistics, pp.80–93. Available from: https://doi.org/10.18653/v1/W19-3509.

Vidgen, Bertie, Nguyen, Dong, Margetts, Helen, Rossini, Patricia, and Tromble, Rebekah, 2021a. Introducing CAD: the contextual abuse dataset. *Proceedings of the 2021 conference of the north american chapter of the association for computational linguistics: human language technologies* [Online]. Online: Association for Computational Linguistics, pp.2289–2303. Available from: https://doi.org/10.18653/v1/2021.naacl-main.182.

Vidgen, Bertie, Thrush, Tristan, Waseem, Zeerak, and Kiela, Douwe, 2021b. Learning from the worst: dynamically generated datasets to improve online hate detection. *Proceedings of the 59th annual meeting of the association for computational linguistics* *and the 11th international joint conference on natural language processing (volume 1: long papers)* [Online]. Online: Association for Computational Linguistics, pp.1667–1682. Available from: https://doi.org/10.18653/v1/2021.acl-long.132.

Wang, Alex, Pruksachatkun, Yada, Nangia, Nikita, Singh, Amanpreet, Michael, Julian, Hill, Felix, Levy, Omer, and Bowman, Samuel R., 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. *Corr* [Online], abs/1905.00537. arXiv: 1905.00537. Available from: http://arxiv.org/abs/1905.00537.

Wang, Alex, Singh, Amanpreet, Michael, Julian, Hill, Felix, Levy, Omer, and Bowman, Samuel R., 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. *Corr* [Online], abs/1804.07461. arXiv: 1804.07461. Available from: http://arxiv.org/abs/1804.07461.

Waseem, Zeerak, 2016. Are you a racist or am I seeing things? annotator influence on hate speech detection on Twitter. *Proceedings of the first workshop on NLP and computational social science* [Online]. Austin, Texas: Association for Computational Linguistics, pp.138–142. Available from: https://doi.org/10.18653/v1/W16-5618.

Waseem, Zeerak, Davidson, Thomas, Warmsley, Dana, and Weber, Ingmar, 2017. Understanding abuse: a typology of abusive language detection subtasks. *Proceedings of the first workshop on abusive language online* [Online]. Vancouver, BC, Canada: Association for Computational Linguistics, pp.78–84. Available from: https://doi.org/10.18653/v1/W17-3012.

Waseem, Zeerak, Thorne, James, and Bingel, Joachim, 2018. Bridging the gaps: multi task learning for domain transfer of hate speech detection [Online]. In: pp.29–55. Available from: https://doi.org/10.1007/978-3-319-78583-7_3.

Yang, Linyi, Zhang, Shuibai, Qin, Libo, Li, Yafu, Wang, Yidong, Liu, Hanmeng, Wang, Jindong, Xie, Xing, and Zhang, Yue, 2022. *Glue-x: evaluating natural language understanding models from an out-of-distribution generalization perspective*. arXiv: 2211.08073 [cs.CL].

Yin, Wenjie and Zubiaga, Arkaitz, 2021. Towards generalisable hate speech detection: a review on obstacles and solutions. *Corr* [Online], abs/2102.08886. arXiv: 2102.08886. Available from: https://arxiv.org/abs/2102.08886.

Zampieri, Marcos, Nakov, Preslav, Rosenthal, Sara, Atanasova, Pepa, Karadzhov, Georgi, Mubarak, Hamdy, Derczynski, Leon, Pitenis, Zeses, and Çöltekin, Çağrı, 2020. SemEval-2020 task 12: multilingual offensive language identification in social media (OffensEval 2020). *Proceedings of the fourteenth workshop on semantic evaluation* [Online]. Barcelona (online): International Committee for Computational Linguistics, pp.1425–1447. Available from: https://doi.org/10.18653/v1/2020.semeval-1.188.

Zhang, Wei Emma, Sheng, Quan Z., Alhazmi, Ahoud, and Li, Chenliang, 2020. Adversarial attacks on deep-learning models in natural language processing: a survey. *Acm trans. intell. syst. technol.* [Online], 11(3). Available from: https://doi.org/10.1145/3374217.

Zhu, Yukun, Kiros, Ryan, Zemel, Richard S., Salakhutdinov, Ruslan, Urtasun, Raquel, Torralba, Antonio, and Fidler, Sanja, 2015. Aligning books and movies: towards story-like visual explanations by watching movies and reading books. *Corr* [Online], abs/1506.06724. arXiv: 1506.06724. Available from: http://arxiv.org/abs/1506.06724.

Ziems, Caleb, Chen, Jiaao, Harris, Camille, Anderson, Jessica, and Yang, Diyi, 2022. VALUE: Understanding dialect disparity in NLU. *Proceedings of the 60th annual meeting of the association for computational linguistics (volume 1: long papers)* [Online]. Dublin, Ireland: Association for Computational Linguistics, pp.3701–3720. Available from: https://doi.org/10.18653/v1/2022.acl-long.258.

Appendix A

Experimental Setup - Details

A.1 Language Models

Model	Parameter	Value
BERT-base-cased	layers	12
	hidden size	768
	attention heads	12
	total parameters	110M
HateBert	layers	12
	hidden size	768
	attention heads	12
	total parameters	110M
BERT-Medium	layers	8
	hidden size	512
	attention heads	12
	total parameters	41.7M
RoBERTa	layers	24
	hidden size	1024
	attention heads	16
	total parameters	355M

Table A.1: Parameters and sizes of language models.

Hyperparameter	Value
batch size	4 (biggest possible)
early stopping	after 5 epochs
maximum epochs	10 (20 for the larger RoBERTa models)
optimizer	AdamW
learning rate	2e-5
adam epsilon	1e-8
scheduling	linear schedule with warmup
warm up steps	0
random seeds	42, 55, 83
max. sequence length	512

Table A.2: Hyperparameters for finetuning the language models adopted from Caselli et al. (2021).

A.2 Clustering

Parameter	Value
n clusters	3-50
n initializations with different centroids	10
max. iterations for a run	300
random state	42, 62, 82
algorithm	LLoyd

Table A.3: K-Means hyperparameters

Appendix B

Detailed Results

This chapter of the appendix presents some detailed results for the different data splitting methods.

B.1 Development Dataset

The following results are achieved by models trained on the development dataset (Qian et al., 2019).

B.1.1 Subset-Sum-Split on Pretrained Embeddings

model	subset-split	valid acc	test acc	f1	roc auc
BERT-base	random	91.69 ± 0.07	91.25 ± 0.11	81.96 ± 0.5	94.35 ± 0.18
	full dims	92.18 ± 0.15	89.54 ± 0.15	78.4 ± 0.32	91.45 ± 0.15
	umap 10	92.0 ± 0.08	89.93 ± 0.52	78.05 ± 0.97	93.51 ± 0.44
	umap 50	92.42 ± 0.23	90.53 ± 0.88	79.69 ± 1.8	93.31 ± 0.79
	umap 200	92.07 ± 0.19	90.78 ± 0.84	80.07 ± 1.68	93.98 ± 0.62

Continued on next page

Table B.1 – continued from previous page

model	subset-split	valid acc	test acc	f1	roc auc
Hate-Bert	random	92.02 ± 0.07	91.51 ± 0.13	82.34 ± 0.59	95.14 ± 0.15
	full dims	92.33 ± 0.08	91.04 ± 0.27	81.03 ± 0.54	94.24 ± 0.17
	umap 10	91.79 ± 0.15	91.6 ± 0.45	82.39 ± 1.0	94.45 ± 0.5
	umap 50	91.89 ± 0.16	92.62 ± 0.44	84.61 ± 0.94	95.56 ± 0.32
	umap 200	92.43 ± 0.19	92.42 ± 0.71	83.74 ± 1.65	95.58 ± 0.35
BERT-medium	random	91.84 ± 0.07	91.2 ± 0.15	81.58 ± 0.66	94.45 ± 0.16
	full dims	92.86 ± 0.32	86.66 ± 0.58	72.24 ± 1.13	90.23 ± 0.38
	umap 10	92.08 ± 0.2	90.89 ± 0.35	80.48 ± 0.76	94.09 ± 0.33
	umap 50	91.82 ± 0.07	89.54 ± 0.4	77.92 ± 0.81	93.12 ± 0.13
	umap 200	92.01 ± 0.17	88.64 ± 1.28	75.51 ± 2.64	92.33 ± 1.04
Ro-BERTa	random	91.8 ± 0.09	91.37 ± 0.16	82.15 ± 0.61	94.32 ± 0.21
	full dims	92.36 ± 0.2	90.83 ± 0.27	80.67 ± 0.46	94.02 ± 0.3
	umap 10	92.17 ± 0.09	89.65 ± 0.41	78.15 ± 1.06	93.3 ± 0.26
	umap 50	92.08 ± 0.18	89.9 ± 0.24	78.13 ± 0.9	93.11 ± 0.22
	umap 200	91.42 ± 0.13	90.84 ± 0.85	78.81 ± 2.17	93.37 ± 0.63

Table B.1: Results on the development dataset (Qian et al., 2019) for the subset-sum-split obtained with **pretrained-embeddings**. The hidden representations used for this split are either used with their original dimensionality (*full dims*), or with decreased dimensions by UMAP. Note that there is no dimensionality reduction by a bottleneck, as this would have to be trained before.

B.1.2 Closest-Split on Pretrained Embeddings

model	closest-split	valid acc	test acc	f1	roc auc
BERT-base	random	91.69 ± 0.07	91.25 ± 0.11	81.96 ± 0.5	94.35 ± 0.18
	full dims	92.24 ± 0.12	83.69 ± 0.1	65.41 ± 0.35	86.89 ± 0.2
	umap 10	91.09 ± 0.19	93.18 ± 0.15	85.71 ± 0.31	95.36 ± 0.2
	umap 50	91.86 ± 0.11	93.82 ± 0.08	86.54 ± 0.17	95.82 ± 0.12
	umap 200	91.51 ± 0.1	91.93 ± 0.06	82.31 ± 0.24	94.13 ± 0.23
Hate-Bert	random	92.02 ± 0.07	91.51 ± 0.13	82.34 ± 0.59	95.14 ± 0.15
	full dims	92.33 ± 0.23	90.77 ± 0.21	80.74 ± 0.59	93.92 ± 0.54
	umap 10	92.37 ± 0.06	92.01 ± 0.12	82.81 ± 0.3	95.79 ± 0.08
	umap 50	92.35 ± 0.37	89.79 ± 0.5	76.85 ± 1.46	93.7 ± 0.05
	umap 200	91.77 ± 0.05	90.81 ± 0.1	80.52 ± 0.38	93.44 ± 0.79
BERT-medium	random	91.84 ± 0.07	91.2 ± 0.15	81.58 ± 0.66	94.45 ± 0.16
	full dims	92.79 ± 0.12	80.44 ± 0.66	62.56 ± 0.71	85.79 ± 0.23
	umap 10	91.66 ± 0.15	93.32 ± 0.03	85.64 ± 0.1	95.91 ± 0.09
	umap 50	91.49 ± 0.11	94.78 ± 0.11	88.74 ± 0.24	96.68 ± 0.13
	umap 200	91.5 ± 0.12	93.22 ± 0.43	85.44 ± 0.82	95.32 ± 0.34
Ro-BERTa	random	91.8 ± 0.09	91.37 ± 0.16	82.15 ± 0.61	94.32 ± 0.21
	full dims	92.09 ± 0.1	93.71 ± 0.13	86.73 ± 0.18	95.72 ± 0.11
	umap 10	91.72 ± 0.07	84.84 ± 0.17	69.05 ± 0.9	88.87 ± 0.27
	umap 50	90.93 ± 0.07	91.1 ± 0.09	81.37 ± 0.16	94.86 ± 0.1
	umap 200	91.87 ± 0.26	91.36 ± 1.4	81.42 ± 2.99	93.68 ± 1.2

Table B.2: Results on the development dataset (Qian et al., 2019) for the closest-split obtained with **pretrained-embeddings**. The hidden representations used for this split are either used with their original dimensionality (*full dims*), or with decreased dimensions by UMAP. Note that there is no dimensionality reduction by a bottleneck, as this would have to be trained before.

B.2 Test Dataset

The following results are achieved by models trained on the test dataset (Mathew et al., 2020).

model	subset-split	valid acc	test acc	f1	roc auc
BERT-base	random	67.66 ± 0.31	68.25 ± 0.28	66.0 ± 0.36	83.6 ± 0.14
	full dims	69.8 ± 0.95	47.32 ± 1.43	43.85 ± 1.2	63.33 ± 1.08
	bottlen. 50	70.95 ± 0.24	51.33 ± 3.09	49.08 ± 3.07	66.89 ± 2.55
	umap 50	68.44 ± 0.38	60.24 ± 2.29	57.51 ± 2.78	75.1 ± 2.12
BERT-medium	random	62.46 ± 0.49	62.85 ± 0.42	60.18 ± 0.42	79.98 ± 0.34
	full dims	66.47 ± 0.27	56.01 ± 3.53	53.09 ± 3.55	70.83 ± 2.72
	bottlen. 50	66.74 ± 0.37	49.34 ± 1.06	47.08 ± 1.08	65.65 ± 0.79
	umap 50	65.24 ± 0.66	54.57 ± 0.83	51.74 ± 1.4	71.99 ± 0.77
Hate-Bert	random	67.91 ± 0.32	68.51 ± 0.28	66.25 ± 0.35	84.46 ± 0.15
	full dims	70.11 ± 0.53	50.76 ± 1.9	50.21 ± 1.84	69.45 ± 1.72
	bottlen. 50	69.77 ± 0.53	51.46 ± 2.64	50.79 ± 2.58	68.42 ± 2.05
	umap 50	70.85 ± 0.21	56.66 ± 0.77	51.62 ± 0.88	73.17 ± 0.33
Ro-BERTa	random	66.45 ± 0.51	66.4 ± 0.56	64.1 ± 0.9	82.19 ± 0.61
	full dims	67.24 ± 2.61	47.67 ± 0.92	43.71 ± 1.33	63.94 ± 0.95
	bottlen. 50	67.49 ± 2.6	45.57 ± 1.54	42.58 ± 1.77	63.53 ± 1.42
	umap 50	67.76 ± 0.35	58.12 ± 1.46	55.42 ± 1.21	72.72 ± 1.12

Table B.3: Results on the test dataset (Mathew et al., 2020) for the subset-sum-split obtained with finetuned-embeddings. The hidden representations used for this split are either used with their original dimensionality (*full dims*), or with decreased dimensions by either UMAP or an implemented bottleneck (*bottlen.*).

model	closest-split	valid acc	test acc	f1	roc auc
BERT-base	random	67.66 ± 0.31	68.25 ± 0.28	66.0 ± 0.36	83.6 ± 0.14
	full dims	72.48 ± 0.43	36.65 ± 1.51	27.78 ± 1.17	62.54 ± 2.81
	bottlen. 50	70.22 ± 1.56	29.02 ± 0.66	26.54 ± 0.72	45.81 ± 0.75
	umap 50	71.8 ± 0.23	37.93 ± 1.33	27.82 ± 2.43	64.33 ± 2.22
BERT-medium	random	62.46 ± 0.49	62.85 ± 0.42	60.18 ± 0.42	79.98 ± 0.34
	full dims	67.21 ± 0.59	31.02 ± 0.23	23.51 ± 1.54	58.28 ± 2.66
	bottlen. 50	66.4 ± 0.67	28.17 ± 0.54	22.54 ± 1.02	45.92 ± 0.28
	umap 50	66.99 ± 0.23	27.91 ± 0.67	26.28 ± 0.49	49.32 ± 1.21
Hate-Bert	random	67.91 ± 0.32	68.51 ± 0.28	66.25 ± 0.35	84.46 ± 0.15
	full dims	72.37 ± 0.25	41.34 ± 0.3	31.55 ± 1.08	61.05 ± 0.42
	bottlen. 50	72.76 ± 0.3	30.14 ± 0.33	21.95 ± 0.35	49.47 ± 0.61
	umap 50	72.89 ± 0.3	40.57 ± 0.44	29.77 ± 1.07	58.25 ± 0.35
Ro-BERTa	random	66.45 ± 0.51	66.4 ± 0.56	64.1 ± 0.9	82.19 ± 0.61
	full dims	71.13 ± 0.58	38.3 ± 0.4	32.89 ± 1.25	57.11 ± 0.52
	bottlen. 50	71.74 ± 0.42	33.13 ± 0.93	27.74 ± 0.68	51.08 ± 2.24
	umap 50	70.5 ± 0.57	34.91 ± 1.72	24.27 ± 0.57	58.48 ± 1.57

Table B.4: Results on the test dataset (Mathew et al., 2020) for the closest-split obtained with finetuned-embeddings. The hidden representations used for this split are either used with their original dimensionality (*full dims*), or with decreased dimensions by either UMAP or an implemented bottleneck (*bottlen.*).

Appendix C

Analysis

C.1 Methodology

Word Overlap: Following the word overlap algorithm in Elangovan, He, and Verspoor (2021), the word overlap o_i for a given test example $test_i$ is the word overlap with the most similar training example $train_k$. The word overlap of the whole test set is then the average over the word overlap of the test examples o_i. For this computation, examples are represented as a vector with unigram counts (ignoring stopwords), and similarity is computed as the cosine similarity.

Rare Word Count: Rare words are defined following the definition of Godbole and Jia (2022): Rare words are words that are not common (i.e. occur at most once per million words) and are not misspelled (i.e. appear in the word list of common words[1]). For word frequency statistics, Godbole and Jia (2022) rely on Brysbaert and New (2009). Alternatively, this work also uses the word frequencies more recently collected by Speer (2022).

[1](https://github.com/dwyl/english-words)