
Interpretability-informed Ensemble Weighting: Final Report

Group: 070 (s1851999, s2436336, s2328524)

Abstract

Ensembles of deep neural nets, and interpretability methods, are both well established in the field of text classification. They respectively provide improved performance on the classification task, and allow humans to understand the model decision. In our project we present a new ensemble technique which is itself based on local interpretability measures from each of the constituent models. To test this method, we compare the performance of this ‘interpretability weighted’ ensemble against a majority-vote baseline. We provide a detailed analysis of interpretability-informed ensembles and find that these models gain a small (although statistically insignificant) improvement on out-of-domain data in comparison to our baselines, while being easier to explain than the other ensembles.

1. Introduction

Model ensembling (Dietterich, 2000; Hansen & Salamon, 1990) is a machine learning technique that combines several base models in order to produce a single superior predictive model, significantly increasing the accuracy of results. Use of ensemble methods is widespread across machine learning, with increased popularity in Natural Language Processing (NLP) in recent years (Copara et al., 2020b; Dadu et al., 2020; Briskilal & Subalalitha, 2022). Concern arose that such models are hard to interpret - while language models are often considered to be ‘black box’, ensembling makes this even harder.

In this project, we combine interpretability methods and ensembling methods with the aims to (1) improve the performance of ensembling and (2) make ensembling more interpretable.

To achieve this, we re-weight the contribution of each model to the overall ensemble according to agreement of scores obtained from interpretability methods. In other words, clusters of models which share the same underlying reasoning for their predictions (indicated by displaying similar interpretability scores) carry more weight, whilst ‘outlier’ models (in terms of interpretability) carry less weight. We focus on creating such an ensemble model and compare its prediction accuracy against standard benchmarks. Moreover, we discuss the benefits of this method with respect to the interpretability of the ensemble prediction in comparison to other ensemble methods.

We test this method on two tasks, hate speech detection and the grammaticality of sentences. We train binary text classification models on a labelled corpus of user posts sourced from a white supremacy forum (de Gibert et al., 2018) and a CoLA dataset (Warstadt et al., 2018) respectively.

To test and analyse our proposed method, we follow the following steps:

1. **Constituent models:** we fine-tune multiple models on the same task, using different architectures such as BERT-medium (Bhargava et al., 2021) and RoBERTa (Liu et al., 2019), repeated multiple times using different random initialisations. This gives a set of diverse models and initial baselines.
2. **Ensemble benchmarks:** we combine the constituent models using standard ensembling approaches to obtain a set of benchmark ensemble models.
3. **Interpretability methods:** we implement a range of local interpretability measures - SHAP (Lundberg & Lee, 2017), LIME (Ribeiro et al., 2016), (self-) attention (Vaswani et al., 2017), and Integrated Gradients (Sundararajan et al., 2017) - and apply them to our set of model predictions.
4. **Interpretability-informed ensemble weighting:** we use the outputs of (2) and (3) to implement our proposed method via interpretability-measure-agreement.

We find that our method, interpretability-informed ensembles, achieves equal or better out-of-domain performance compared to standard ensemble approaches while being easier to interpret.

The report is structured as follows: the dataset and task are described in Section 2, while Section 3 presents the benchmark ensemble techniques and required interpretability methods. Our new method is introduced in Section 4. Experiment results and analysis are presented in Section 5, Section 6 discusses related work, before concluding in Section 7. Our code and results are available on Github¹.

2. Dataset and task

To evaluate our proposed method, the interpretability-informed ensemble, we test it on two different (binary) classification tasks. The first task is to predict whether an online forum post contains hate speech, the second task is

¹https://github.com/dolmas1/MLP_Group_Project

to predict whether or not a sentence is grammatical (CoLA). A successful hate speech detection system (or grammaticality detection system) requires both high recall (rarely miss true occurrences) as well as high precision (low false positive rate), so we will assess performance in terms of f1-score - a standard metric defined as the harmonic mean of precision and recall.

2.1. Hate Speech

We use a dataset by [de Gibert et al. \(2018\)](#) consisting of 10,568 sentences collected from Stormfront, a white supremacist forum. Each sentence has been manually labelled as ‘hate speech’ or ‘non-hate speech’. [de Gibert et al. \(2018\)](#) take a subset of these posts to obtain a balanced sample on which to run their models, resulting in 2,392 sentences split equally between hate and non-hate examples. To allow direct comparison of results we use the same balanced sample, and perform a standard train/dev/test split. We obtain 1914/239/239 examples respectively. Moreover, we perform some simple pre-processing steps (removal of URLs) on the data to reduce tokenisation errors. An example from the dataset can be found below:

Labelled hate: *Compared to the stunningly beautiful white Swedish females she is just a barfy mixed breed dog*

To test the robustness of our method, we also evaluate performance on a test set from a different source. We use the Twitter dataset *Hate Speech / Offensive Speech in the US 2020 Elections* ([Grimminger & Klinger, 2021](#)), which includes 600 labelled test sentences. Again, pre-processing was performed to remove URLs and emojis.

2.2. CoLA

The second task we evaluate on is to predict whether a sentence is grammatical or not. We use the CoLA dataset by [Warstadt et al. \(2018\)](#). It contains 10,657 sentences, collected from 23 publications and manually annotated according to whether the sentences are grammatical. Around 30% of the data are labelled positively (i.e. ungrammatical). We use the same train/dev/test split proposed by the authors, resulting in 8,551 train, 527 dev, and 516 test sentences.

Importantly, the split is made in a way that the train and dev set contain sentences from 17 publication sources, whilst the test set contains the other 6 sources. This makes the test set already *out-of-domain*, so there is no need to test our method on another separate test set as before. An example from the dataset is as follows:

Labelled as not grammatical: *John ate dinner but I don't know who.*

3. Methodology

In Section 3.1 we provide a discussion of three standard approaches for creating an ensemble (which we implement as a baseline). In Section 3.2, we then explain the four local-interpretability methods used in this project.

3.1. Ensembling

Ensemble methods are a set of techniques whereby multiple independent models are trained to perform the same task (in our case, text classification). Throughout this report we will use the term *individual* or *constituent* model to refer to each of the models, to distinguish them from the additional (ensemble) models which will be produced from them.

During inference, the predictions from each of the models are combined to make a single prediction for each test observation. If a diverse set of constituent models are used, the classification errors should be uncorrelated enough such that the ensemble prediction is more accurate than any single model.

We present a description of the three ensemble approaches we implement as benchmarks. A brief overview of some alternative techniques can be found in Section 6.

3.1.1. MAJORITY VOTING

Majority voting is the simplest possible approach for ensembling a set of model predictions to make a single classification decision: the ensemble prediction will be the majority decision across each of the constituent models. For example, if six out of nine models predict ‘hate speech’, the ensemble prediction will also be ‘hate speech’ (we use an odd number of constituent models to avoid ties). This has the advantage of not requiring a separate validation set in order to tune the ensembling approach, and will act as our baseline against which other ensembling approaches will be tested.

In the wider NLP literature, majority voting has been prominently used for Named Entity Recognition with an ensemble of BERT models ([Copara et al., 2020b](#); [Knafo et al., 2020](#), NER). One slight modification is to use a threshold for the majority vote ([Copara et al., 2020a](#)), also in NER.

3.1.2. STRAIGHT AVERAGE PREDICTION

If each individual model outputs a predicted probability rather than a binary classification label, another approach is to calculate the average of these probabilities before applying a classification threshold. This approach is conceptually similar to majority voting and also commonly used ([Kanakaraj & Guddeti, 2015](#)). *Straight average* ensembles will act as our second baseline.

3.1.3. NETWORK WEIGHT AVERAGING

Network weight averaging is specific to deep neural networks. If the constituent models share an identical architecture, we can combine them to create a single model (again with the same architecture) whose weight parameters are the average of the weights associated with each constituent model ([Utans, 1996](#)). Our methodology uses three different architectures of constituent models (all deep networks), each of which is repeatedly fine-tuned. We shall implement *network weight averaging* as our third baseline: first by weight-averaging the set of models associated with each

architecture, then majority voting across the three resulting models.

3.2. Interpretability Methods

This section presents a description of four interpretability methods: LIME, SHAP, attention, and Integrated Gradients. When we use each model to make class predictions for a given test sentence, each interpretability method will produce a set of scalar importance scores attached to each input token. These are known as *local* interpretability measures, they explain which particular input features (tokens) led the model to make its prediction for this particular observation.

As well as being vital tools in their own right for implementing text classification systems (both for debugging purposes and real-world end users), comparing these measures will allow us to assess how much two models agree in terms of the reasoning behind their decisions. These *agreement scores* will ultimately be used to re-weight the constituent models in our new ensembling technique described in Section 4. All four of the methods will be used. Our implementations are adapted from LimeTextExplainer², Captum integrated gradients³ and the SHAP library⁴. The attention weights are accessed directly from the transformer models.

3.2.1. LIME

Local interpretable model-agnostic explanations (Ribeiro et al., 2016, LIME) are based on a *local surrogate model*. Surrogate models are trained to approximate the predictions of the underlying model, which is treated as a ‘black box’, in nearby feature-space. Instead of aiming for global explanations, LIME focuses on training a separate local surrogate model (in this case a ridge regression) to explain each individual prediction. To do this for text data, LIME generates a synthetic dataset consisting of perturbed samples (perturbed word order and dropped words in sentences) and the corresponding predictions of the original model. It then compares the outputs obtained with the perturbed data with the original output. Intuitively, if the output has significantly changed after perturbing a specific input, this input must be important and gets a high score. Conversely, if perturbing an input does not significantly change the output then a low importance score is assigned. Since we have limited GPU memory, we restrict the sample size of each dataset to 30, however it is possible that better results could be obtained with a higher sample number.

3.2.2. SHAP

SHapley Additive exPlanations (Lundberg & Lee, 2017, SHAP) is a method to explain predictions for an individual sentence by computing the contribution of each word to the prediction. This is done by computing so-called Shapley values (Lipovetsky & Conklin, 2001; Štrumbelj & Kononenko, 2014) originally developed for coalitional

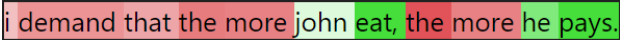


Figure 1. Example visualisation of LIME output calculated on an sample sentence from the CoLA dataset. A strong green colouring indicates the model placing high importance on a particular token when making an ‘ungrammatical’ prediction. In this example, it is clear that the model has correctly identified a grammatical agreement error involving the verbs ‘eat’ and ‘pays’. Similar visualisations could also be produced for any of the following interpretability methods.

game theory. Those values indicate how to fairly distribute the the prediction among the features and therefore which feature was important for the prediction. Shapley values behave slightly differently compared to LIME scores, due to their *additive* property - the sum of the SHAP scores across all tokens is equal to the overall model prediction. For this reason, both methods will prove useful for our interpretability-informed ensembling.

3.2.3. ATTENTION

Our third interpretability method is especially useful for NLP tasks and based on (self-) attention (Vaswani et al., 2017), a widely used technique in modern large language models. Intuitively, self-attention calculates the similarity between each word of the sentence with every other word (including itself). To obtain an importance score for each word, we first take the self-attention scores between each word and the CLS (class) token. We then take the sum over all heads and layers as proposed in Luo et al. (2020):

$$attn_score(CLS, tok) = \sum_{layers} \sum_{heads} attention(CLS, tok)$$

3.2.4. INTEGRATED GRADIENTS

The last interpretability method we explore is Integrated Gradients (Sundararajan et al., 2017). The goal of Integrated Gradients is to attribute a value to each input feature, based on how much the input contributes an increment to the final prediction value. The Integrated Gradients method starts with a so-called baseline, i.e. a starting input containing no information useful for the model. For language tasks, this baseline is the empty string containing start and end of sentence tokens only. We then incrementally add each token of the original sentence to the sequence. At each step, the prediction and gradients of the model are calculated. “Integrated gradients” are the sum of these gradients.

4. New Approach: Interpretability-informed Ensembles

The main contribution of our project is our newly developed method: Interpretability-informed ensembles. In the following, we provide an overview (Section 4.1) alongside with the motivation (Section 4.2) and a precise definition of the method (Section 4.3).

²<https://github.com/marcotcr/lime>

³https://captum.ai/docs/extension/integrated_gradients

⁴<https://github.com/slundberg/shap>

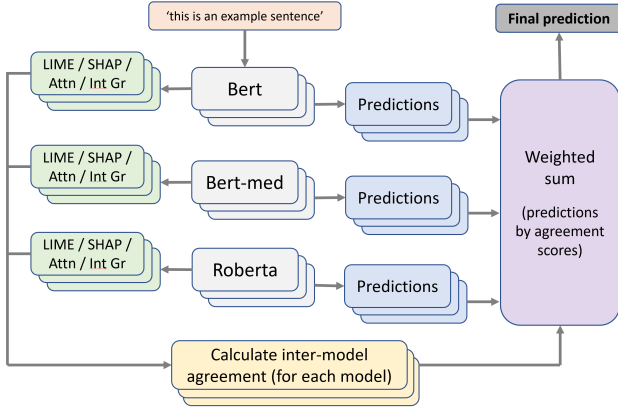


Figure 2. High level illustration of interpretability-informed ensembles.

4.1. Overview

We have developed an ensemble technique to compete against the baseline approaches described above - see Figure 2 for a high level visual summary and Section 5 for the details of the constituent models of the ensemble.

The core idea is that instead of directly using the spread of predicted model scores (and reducing to a single prediction via majority vote or stacking methods), we first use the set of **local interpretability measures** coming from each of the constituent models to form an alternative description of model diversity.

As described in Section 3.2, these measures comprise of scalars associated with each input token for each test sentence, describing the contribution of each token to either a positive (hate speech) or negative (non-hate speech) class prediction.

Using these values, we calculate the degree of pairwise agreement (cosine similarity) between models in terms of interpretability scores. Our algorithm reduces the weighting of any model whose interpretation significantly diverges from the average, relying instead on the remaining models to form our final ensemble prediction.

4.2. Desired benefits

Our goal is that interpretability-informed ensembles form a more principled approach to selecting the ‘best’ prediction from the constituent models, compared to simple majority voting. Note that selecting the model that is most similar to the other constituent models in terms of model interpretability *does not* necessarily correspond to selecting the mean or median of the model *predictions*.

This point is crucial - for example, it may be that a cluster of three models correctly predict ‘hate speech’ for a given sentence, and also all agree on the reason (e.g. high interpretability scores for one of the tokens, which may be a racial slur). The remaining six models may all predict ‘non-hate speech’, but for a variety of different reasons. In

this case, our proposed method would make the correct prediction whilst simple majority voting would fail. In Section 5.4 we provide some concrete examples from the dataset to support this line of reasoning.

An additional advantage emerges when we wish to explain the ensemble predictions for a particular example. For example, an ensemble-based hate speech detection system may flag a particular user to the police. In order to assist the investigation it would be helpful for the officer to be quickly told the system’s reason for making the prediction. To the authors’ knowledge, the only way to generate model explanations for an ensemble model is to first generate explanations for each constituent model, and inspect them in aggregate.

For most ensemble methods, this may not produce a single clear reason - the constituent models may present contradictory explanations. However for interpretability-informed ensembles, we ensure by design that all models in the final ensemble agree with each other. We can ignore contradictory explanations from the outlier models, so interpretation is straightforward.

4.3. Precise definition

The pseudo-code shown in Algorithm 1 defines precisely how we calculate our interpretability-informed ensemble predictions. Note that we retain a choice of methodology at two important points:

- Defining **which interpretability measure to use** - in Section 5.2 we will analyse a range of interpretability ensembles formed either using only LIME scores, only SHAP, only Attention, only Integrated Gradients, or a combination (where we take an average of the model agreement scores from each method). In Algorithm 1, lines 6, 12, 14, 18 must be edited to instead calculate interpretability ensembles for a single measure.
- After calculating the overall agreement score for each constituent model, the final ensemble prediction needs to be determined. We test both **modal voting**, where only the single constituent model with highest agreement score is kept, or **weighted average** - where all models are kept and a weighted sum of predicted probabilities (with respect to agreement score) is taken. In both cases, a classification threshold of 0.5 is applied.

Multiple interpretability measures are used to cover different notions of explainability - the hope is that by using a combination of LIME, SHAP, attention and Integrated Gradients we obtain a more holistic view of model similarity which may perform better than using a single measure. Note that we average across the interpretability measures only *after* calculating an inter-model similarity score for each method. This avoids two differing measures cancelling each other out before the similarity calculation is performed.

One important implementation detail is that of **different tokenisers**. In practice, the BERT and RoBERTa tokenisers


	S	h	e	'	s		r	e	a	d	i	n	g
LIME for BERT:	0.6		0.3	0.8				0.9					
LIME for RoBERTa:	0.3		0.6				0.2			0.4			
 UNIFICATION													
LIME for BERT:	0.6		0.3 + 0.8					0.9					
LIME for RoBERTa:	0.3		0.6					0.2 + 0.4					

Figure 3. Simple example of unification across tokens and associated interpretability scores, where the input sentence *she's reading* has been tokenised in two different ways

will produce a different set of tokens for the same input string. Because the interpretability measures act on this token array rather than the original string, this means that each of the model interpretability arrays may have different length - preventing computation of cosine similarity.

We therefore run a *token unification* algorithm (line 9 in Algorithm 1, details omitted for clarity). Essentially we compare the token arrays across models, iteratively merge adjacent tokens where necessary, and sum the associated interpretability scores. A toy example is shown in Figure 3.

Algorithm 1 Interpretability ensembles

```

1: Input:  $n$  constituent models  $\{m_i\}_{i=1}^n$ 
2: Input: single test sentence  $S$ 
3:
4: for  $i = 1$  to  $n$  do
5:   compute model predicted probability:  $p_i = m_i(S)$ 
6:   calc. interp. scores  $\{I_i^{LIME}, I_i^{SHAP}, I_i^{Attn}, I_i^{IntGr}\}$ 
7: end for
8:
9: run procedure: token unification
10:
11: for  $i = 1$  to  $n$  do
12:   init agreement scores:  $a_i^{LIME}, a_i^{SHAP}, a_i^{Attn}, a_i^{IntGr} = 0$ 
13:   for  $j = 1$  to  $n$  do
14:     for  $method$  in  $\{LIME, SHAP, Attn, IntGr\}$  do
15:        $a_i^{method} += \text{cosine\_similarity}(I_i^{method}, I_j^{method})$ 
16:     end for
17:   end for
18:    $a_i^{comb} = (a_i^{LIME} + a_i^{SHAP} + a_i^{Attn} + a_i^{IntGr})/4$ 
19: end for
20:
21: if  $method = \text{weighted average}$  then
22:    $\bar{p} = \sum_i a_i^{comb} p_i / \sum_i a_i^{comb}$ 
23: else if  $method = \text{modal vote}$  then
24:    $\bar{p} = \underset{p_i}{\text{argmax}} a_i^{comb}$ 
25: end if
26:
27: if  $\bar{p} > 0.5$  then
28:   return 1
29: else
30:   return 0
31: end if

```

5. Experiments

In this section we present our fine-tuned constituent models (Section 5.1) along with a range of baseline and interpretability ensembles. In Section 5.2, our full range of models are compared on both the hate speech and CoLA dev sets. We then choose which models to progress to the final test sets (which include out-of-domain data), with results and discussion found in Section 5.3. Two specific examples are analysed in detail for an illustrated motivation of the benefits of interpretability ensembles (Section 5.4).

5.1. Our constituent models

We have chosen to use BERT-base (Devlin et al., 2018), BERT-medium (Bhargava et al., 2021), and RoBERTa-base (Liu et al., 2019) as our set of three base model ensemble constituents. These models are particularly well suited to our project: pre-trained weights are easily available and fine-tuning is possible on the compute hardware we have available.

Our choice is supported by work such as Briskilal & Subalalitha (2022), in which the authors used an ensemble of BERT-base and RoBERTa to solve an idiom detection classification task with higher performance than any single model.

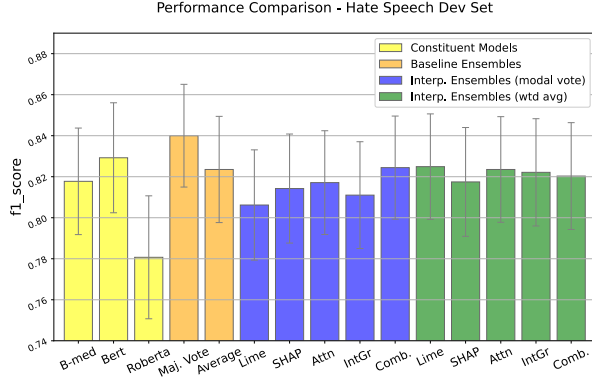
For each of the three models (BERT-base, BERT-medium and RoBERTa-base) we perform fine-tuning on a hate speech dataset (de Gibert et al., 2018) using pre-trained weights taken from HuggingFace and three different random initialisations for each of the base models, giving a total of nine models to use as ensemble constituents. We do the same for the CoLA dataset (Warstadt et al., 2018). Specific hyperparameters are given in Appendix 8. The combination of repeated fine-tuning, as well as diverse model architectures, is designed to give us the maximum amount of model diversity required for successful ensembles.

In Table 1 (equivalently, Figure 4 the yellow bars), we show the precision, recall, and F1-scores for the hate speech task attained by these baseline models on the development set, averaged across the three different initializations. Also shown is the standard deviation, indicating that performance (in terms of F1-score) ranges from around 75% to 85%. This relatively wide range is crucial, as we need a diverse set of constituent models from which to build our ensembles. Achieving F1-scores in the region of 80% is similar to results obtained in de Gibert et al. (2018), as expected.

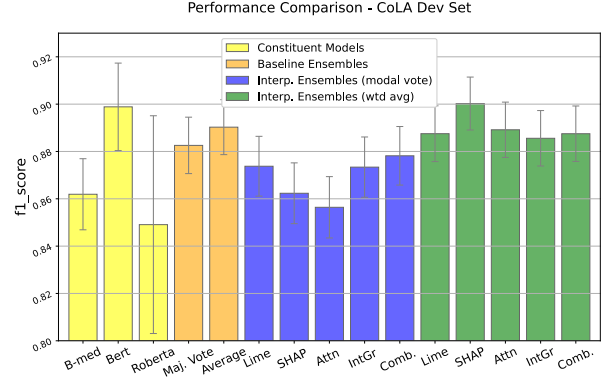
Respectively, results for the CoLA dataset can be found in Table 2, also illustrated in Figure 4. Again, these results are in line with the literature (Warstadt et al., 2018). Performance is broadly good, with some constituent models hitting f1-scores close to 90%.

5.2. Analysis of ensembles on development set

On top of the constituent models mentioned before, we implemented each of the three baseline ensemble approaches outlined in section 3.1. On the hate speech task (see Ta-



(a) Dev set results - hate speech



(b) Dev set results - CoLA

Figure 4. F1-scores on the development set for the different constituent models (yellow), the ensembling baselines (orange), and interpretability ensembles using modal vote (blue) and weighted average (green).

MODEL TYPE	MODEL	PREC.	RECALL	F1-SCORE
CONSTITUENT MODELS (MULT. RANDOM SEEDS)	BERT-MEDIUM	77.37 ±3.81	86.88 ±3.27	81.77 ±2.6
	BERT	83.07 ±3.65	82.94 ±3.76	82.92 ±2.68
	ROBERTA	78.37 ±4.24	78.02 ±4.41	78.07 ±3
BASELINE ENS.	MAJ. VOTE	80.77	87.5	84
	NN WT AVG.	85.96	40.83	55.37
	STRAIGHT AVG.	77.78	87.5	82.35
INTERPRET. ENS. (MODAL VOTE)	LIME	75.36	86.67	80.62
	SHAP	77.44	85.83	81.42
	ATTENTION	76.64	87.5	81.71
	INT GRAD	76.87	85.83	81.1
	COMBINED	76.06	90	82.44
INTERPRET. ENS. (WTD AVG)	LIME	77.37	88.33	82.49
	SHAP	78.03	85.83	81.75
	ATTENTION	77.78	87.5	82.35
	INT GRAD	78.2	86.67	82.21
	COMBINED	77.21	87.5	82.03

Table 1. Hate speech task, dev set results across all models

ble 1), the majority vote and straight average baseline approaches perform well (roughly on par with the best individual model). The network weight averaging baseline has far worse performance, primarily due to low recall. Further analysis suggests that the classification threshold for this model is poorly calibrated: for reasons of brevity we remove it from all subsequent testing.

Following the method described in Section 4, we implemented ten different interpretability ensembles for both the hate speech and CoLA tasks: all five measures (LIME, Shap, Attention, Integrated Gradients and the combined measure), each implemented using modal vote and weighted average. The results tables also include dev set performance metrics for these models. Performance is generally better than individual constituent models, and roughly on par with the baseline ensembles.

At this point we aim to find the most suitable interpretability measure to use in the ensemble calculation. A comparison of results across all five measures determines that the combined interpretability measure performs best for the modal vote ensembles, across both hate speech and CoLA. For weighted average ensembles there is no clear winner - we therefore decide that the *combined interpretability measure* will be our chosen model configuration to carry to the test set.

MODEL TYPE	MODEL	PREC.	RECALL	F1-SCORE
CONSTITUENT MODELS (MULT. RANDOM SEEDS)	BERT-MEDIUM	79.97 ±3.13	93.66 ±2.74	86.19 ±1.5
	BERT	85.87 ±2.33	94.36 ±2.64	89.89 ±1.85
	ROBERTA	74.59 ±7.86	99.27 ±1.12	84.91 ±4.6
BASELINE ENS.	MAJ. VOTE	80.41	97.81	88.26
	STRAIGHT AVG.	81.69	97.81	89.03
INTERPRET. ENS. (MODAL VOTE)	LIME	79.68	96.71	87.38
	SHAP	81.98	90.96	86.23
	ATTENTION	80.48	91.51	85.64
	INT GRAD	82.64	92.6	87.34
	COMBINED	81.8	94.79	87.82
INTERPRET. ENS. (WTD AVG)	LIME	80.86	98.36	88.75
	SHAP	84.41	96.44	90.03
	ATTENTION	81.51	97.81	88.92
	INT GRAD	81.09	97.53	88.56
	COMBINED	81.61	97.26	88.75

Table 2. CoLA task, dev set results across all models

In terms of using the single constituent model with highest overall agreement score (‘modal vote’) in comparison to an interpretability-weighted-average prediction (‘wtd avg’), there is no clear distinction - the wtd avg method performs best on CoLA, but for hate speech the modal vote performs best. We note also that the best *baseline* ensemble method is also task dependent. For this reason, we take as our final models **both** the (combined) modal vote and (combined) weighted average interpretability ensembles.

MODEL TYPE	MODEL	F1-SCORE		
		HATE SPEECH	TWITTER	CoLA
CONSTITUENT MODELS (MULT. RANDOM SEEDS)	BERT-MEDIUM	79.03 \pm 3.07	25.84 \pm 4.45	82.95 \pm 1.57
	BERT	79.94 \pm 2.93	23.17 \pm 2.98	86.67 \pm 1.38
	ROBERTA	75.91 \pm 3.14	26.46 \pm 3.32	83.47 \pm 3.27
BASELINE ENSEMBLES	MAJORITY VOTE	81.08	29.57	85.61
	STRAIGHT AVERAGE	82.63	29.82	85.93
INTERPRET. ENSEMBLES	MODAL VOTE (COMBINED)	78.55	27.03	83.94
	WTD AVG (COMBINED)	81.99	30.5	86.08

Table 3. Test set results across all our datasets

5.3. Test set performance comparison, discussion

Final F1-scores for our chosen models are shown in Table 3, alongside our benchmarks. For the hate speech task we report model performance on the test set associated with our original training data, as well as a separate out-of-domain Twitter dataset (see Section 2.1 - note we do not retrain the models for this). For the CoLA task, the associated test set is already designed as an out-of-domain sample.

As expected, performance drops on the out-of-domain data when compared to the in-domain development sets (especially for the Twitter data).

On the **hate speech** holdout test set, our interpretability ensembles are on par with (or slightly below) the benchmark ensembles. On the **Twitter** dataset, our best model (the weighted average interpretability ensemble) outperforms the highest performing benchmark model (straight average ensemble) by a small amount: F1-score 29.8% vs 29.6%.

On the **CoLA** test set, our weighted average interpretability ensemble again outperforms the straight average baseline ensemble by a small amount: F1-score 86.1% vs 85.9%.

To calculate the significance of these results, we performed random bootstrap evaluations on the Twitter and CoLA test sets. Ten thousand random samples (with replacement) were taken from each test set, and F1-scores recalculated each time. We then inspect the fraction of bootstrap samples for which our model failed to outperform the baseline.

We find that for both the Twitter dataset (p -value = 0.234) and the CoLA test set (p -value = 0.278), the superior performance of our model does not attain statistical significance above the baseline ensemble. However note that as explained in 4.2, we do at least see some ensemble performance benefit *whilst preserving model interpretability*.

It is interesting that our method performed better on out-of-domain data: perhaps interpretability measures display more variability for examples unlike those seen during training. Further analysis could be performed here.

5.4. Justifying examples

We present two concrete examples from the dataset, illustrating how interpretability ensembles can produce more

accurate model predictions than a simple majority vote. One is from the CoLA task, the other from hate speech.

In Figure 6a we represent the predictions from our nine constituent models on an ungrammatical CoLA sentence. Each model is labelled, coloured according to its prediction (grammatical vs ungrammatical), and positioned on a 2D grid according to interpretability measure similarity⁵.

Note that five out of nine models incorrectly deem this sentence as grammatical, but a cluster of four models with high levels of interpretability agreement give the correct prediction. Therefore, our interpretability ensemble correctly classifies this sentence as ungrammatical (the model with highest agreement score - the ‘modal vote’ - is bert_3). Note that in this case, the cluster of similar models are *not* all from the same model family.

An illustration of the interpretability measures themselves is shown in Figure 5. Correct models are seen to place high importance on the sub-phrase ‘some my’ (which is missing the word ‘of’), whilst the other models place higher importance on other parts of the sentence.

Correct?	Model	Explanation
✓	Bert 1	some my jobs are in jeopardy.
✓	Bert 3	some my jobs are in jeopardy.
✓	Bert med 2	some my jobs are in jeopardy.
✓	Bert med 3	some my jobs are in jeopardy.
✗	Roberta 1	some my jobs are in jeopardy.
✗	Roberta 2	some my jobs are in jeopardy.
✗	Roberta 3	some my jobs are in jeopardy.
✗	Bert 2	some my jobs are in jeopardy.
✗	Bert med 1	some my jobs are in jeopardy.

Figure 5. Model interpretability measures for a CoLA example.

Figure 6b also contains a similar representation for an example hate speech sentence. Note that six out of nine

⁵We take the 9x9 matrix of model vs model cosine similarity scores (averaged across all four measures), and apply an MDS algorithm (Mead, 1992) to embed each model into 2D space. Models with similar interpretations are placed closer together.

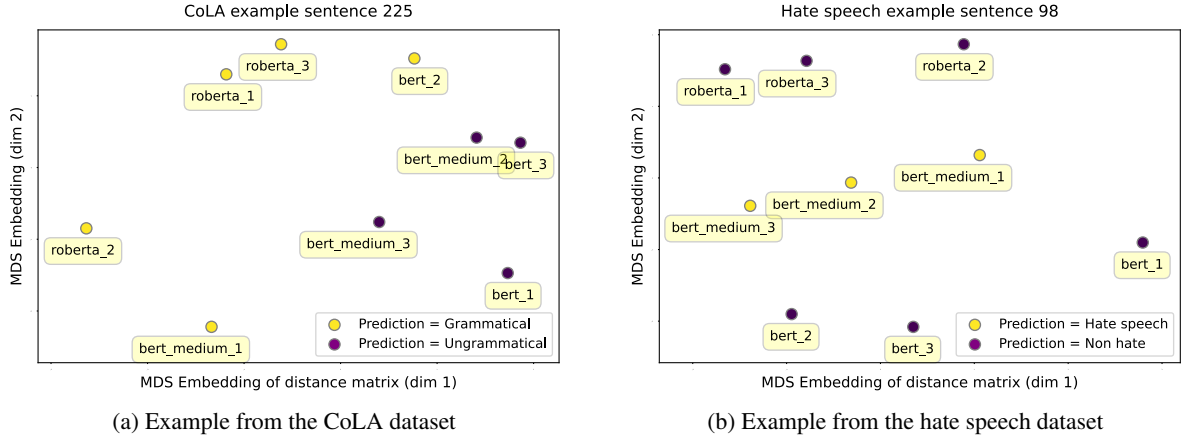


Figure 6. Justifying examples: model predictions, spaced according to interpretability similarity.

models incorrectly predict ‘no-hate’ for this sentence, but the three correct models are closer together in terms of interpretability agreement (the modal vote is bert_medium_2). Therefore, our interpretability ensemble correctly classifies this sentence as hate speech.

6. Related work

Whilst our *interpretability weighted ensemble* extends the concept of averaging predicted probabilities across different models (via weighting by model agreement scores), other methods exist to create weighted ensembles.

6.1. Stacked Models

‘Stacked models’ uses a secondary classifier trained on the set of predicted class probabilities taken from each constituent model. This must be done on a validation dataset - we chose not to implement these as additional benchmarks due to the small size of our validation set. Related work in this area includes [Dadu et al. \(2020\)](#); [Briskilal & Subalalitha \(2022\)](#) who perform linear regression to build a weighted average ensemble for idiom detection. [Risch & Krestel \(2020\)](#) rely on bootstrap aggregating of different language models when performing aggression detection. Other ensemble methods such as naive ensembling, where the class with the highest accumulated probability is chosen ([Pietiläinen & Ji, 2022](#)) are also used for NER.

6.2. Latent Space Averaging

Latent Space Averaging is performed across a set of deep networks which share the same architecture. Latent embeddings are extracted from the penultimate network layer of each constituent model for each test observation. An average of these embeddings is taken (remaining in the same latent space), and the resulting embedding passed through a final classifier. We did not implement this method in our project for reasons of time. Related work in this area includes [Duan et al. \(2021\)](#) who cluster data according to topics and train different LMs before ensembling, show-

ing promising results for sentiment classification and abstractive summarisation. [Huy et al. \(2022\)](#) use siamese networks and learn weights to do the ensemble voting. A completely end-to-end approach was proposed by [Pietiläinen & Ji \(2022\)](#) who concatenate the prediction of the different language models and pass them through a final layer.

6.3. Interpretability of Ensembling

In the medical and chemical domain, when methods such as random forest, extreme randomized trees or boosting are used, interpretability of ensembling involves relying on preserving the feature interpretability of the single models ([Chen et al., 2020](#)). [Cagnini et al. \(2020\)](#) developed an evolutionary algorithm that has properties that simplify interpretability and can be used instead of random forest. [Liang et al. \(2022\)](#) use SHAP ([Lipovetsky & Conklin, 2001](#)) to highlight important features when modelling properties of cement using ensembles.

To the best of our knowledge, in the domain of Natural Language Processing, there is no work on presenting interpretable ensembles. This project aims to suggest a starting point in the discussion.

7. Conclusions

We presented a new technique to build ensembles based on similarity of model explanations. We implemented this new method on deep neural models trained for two text classification tasks, hate speech prediction and linguistic acceptability. We found that when applied to out-of-domain test data, these interpretability ensembles achieve equal-or-better performance in comparison to benchmark ensembles while being (by design) easier to interpret. Despite the extra implementation complexity, we believe this is a promising area for future study - for example by testing on wider set of benchmarks, or carrying out human evaluation of the model explanations.

References

- Bhargava, Prajjwal, Drozd, Aleksandr, and Rogers, Anna. Generalization in nli: Ways (not) to go beyond simple heuristics, 2021.
- Briskilal, J and Subalalitha, C.N. An ensemble model for classifying idioms and literal texts using bert and roberta. *Information Processing & Management*, 59(1):102756, 2022. ISSN 0306-4573. doi: <https://doi.org/10.1016/j.ipm.2021.102756>. URL <https://www.sciencedirect.com/science/article/pii/S0306457321002375>.
- Cagnini, Henry E. L., Freitas, Alex A., and Barros, Rodrigo C. An evolutionary algorithm for learning interpretable ensembles of classifiers. In Cerri, Ricardo and Prati, Ronaldo C. (eds.), *Intelligent Systems*, pp. 18–33, Cham, 2020. Springer International Publishing. ISBN 978-3-030-61377-8.
- Chen, Chia-Hsiu, Tanaka, Kenichi, Kotera, Masaaki, and Funatsu, Kimito. Comparison and improvement of the predictability and interpretability with ensemble learning models in qspr applications. *Journal of Cheminformatics*, 12(1):19, Mar 2020. ISSN 1758-2946. doi: 10.1186/s13321-020-0417-9. URL <https://doi.org/10.1186/s13321-020-0417-9>.
- Copara, Jenny, Knafo, Julien, Naderi, Nona, Moro, Claudia, Ruch, Patrick, and Teodoro, Douglas. Contextualized French language models for biomedical named entity recognition. In *Actes de la 6e conférence conjointe Journées d'Études sur la Parole (JEP, 33e édition), Traitement Automatique des Langues Naturelles (TALN, 27e édition), Rencontre des Étudiants Chercheurs en Informatique pour le Traitement Automatique des Langues (RÉCITAL, 22e édition). Atelier Défi Fouille de Textes*, pp. 36–48, Nancy, France, 6 2020a. ATALA et AFCP. URL <https://aclanthology.org/2020.jeptalnrecital-deft.4>.
- Copara, Jenny, Naderi, Nona, Knafo, Julien, Ruch, Patrick, and Teodoro, Douglas. Named entity recognition in chemical patents using ensemble of contextual language models. *CoRR*, abs/2007.12569, 2020b. URL <https://arxiv.org/abs/2007.12569>.
- Dadu, Tanvi, Pant, Kartikey, and Mamidi, Radhika. Bert-based ensembles for modeling disclosure and support in conversational social media text, 2020.
- de Gibert, Ona, Perez, Naiara, García-Pablos, Aitor, and Cuadros, Montse. Hate speech dataset from a white supremacy forum. In *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*, pp. 11–20, Brussels, Belgium, October 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-5102. URL <https://aclanthology.org/W18-5102>.
- Devlin, Jacob, Chang, Ming-Wei, Lee, Kenton, and Toutanova, Kristina. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. URL <http://arxiv.org/abs/1810.04805>.
- Dietterich, Thomas G. Ensemble methods in machine learning. In *Multiple Classifier Systems*, pp. 1–15, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg. ISBN 978-3-540-45014-6.
- Duan, Zhibin, Zhang, Hao, Wang, Chaojie, Wang, Zhengjue, Chen, Bo, and Zhou, Mingyuan. EnsLM: Ensemble language model for data diversity by semantic clustering. In *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)*, pp. 2954–2967, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.230. URL <https://aclanthology.org/2021.acl-long.230>.
- Grimminger, Lara and Klinger, Roman. Hate towards the political opponent: A Twitter corpus study of the 2020 US elections on the basis of offensive speech and stance detection. In *Proceedings of the Eleventh Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pp. 171–180, Online, April 2021. Association for Computational Linguistics. URL <https://aclanthology.org/2021.wassa-1.18>.
- Hansen, L.K. and Salamon, P. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):993–1001, 1990. doi: 10.1109/34.58871.
- Huy, Ngo Quang, Phuong, Tu Minh, and Bach, Ngo Xuan. Autoencoding language model based ensemble learning for commonsense validation and explanation, 2022.
- Kanakaraj, Monisha and Guddeti, Ram Mohana Reddy. Performance analysis of ensemble methods on twitter sentiment analysis using nlp techniques. In *Proceedings of the 2015 IEEE 9th International Conference on Semantic Computing (IEEE ICSC 2015)*, pp. 169–170, 2015. doi: 10.1109/ICOSC.2015.7050801.
- Knafo, Julien, Naderi, Nona, Copara, Jenny, Teodoro, Douglas, and Ruch, Patrick. BiTeM at WNUT 2020 shared task-1: Named entity recognition over wet lab protocols using an ensemble of contextual language models. In *Proceedings of the Sixth Workshop on Noisy User-generated Text (W-NUT 2020)*, pp. 305–313, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.wnut-1.40. URL <https://aclanthology.org/2020.wnut-1.40>.
- Liang, Minfei, Chang, Ze, Wan, Zhi, Gan, Yidong, Schlangen, Erik, and Šavija, Branko. Interpretable ensemble-machine-learning models for predicting creep behavior of concrete. *Cement and Concrete Composites*, 125:104295, 2022. ISSN 0958-9465. doi: <https://doi.org/10.1016/j.cemconcomp.2021.104295>. URL <https://www.sciencedirect.com/science/article/pii/S0958946521003620>.

-
- Lipovetsky, Stan and Conklin, Michael. Analysis of regression in game theory approach. *Applied Stochastic Models in Business and Industry*, 17(4):319–330, 2001. doi: <https://doi.org/10.1002/asmb.446>. URL <https://onlinelibrary.wiley.com/doi/abs/10.1002/asmb.446>.
- Liu, Yinhan, Ott, Myle, Goyal, Naman, Du, Jingfei, Joshi, Mandar, Chen, Danqi, Levy, Omer, Lewis, Mike, Zettlemoyer, Luke, and Stoyanov, Veselin. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019. URL <http://arxiv.org/abs/1907.11692>.
- Lundberg, Scott M and Lee, Su-In. A unified approach to interpreting model predictions. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30*, pp. 4765–4774. Curran Associates, Inc., 2017. URL <http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf>.
- Luo, Xiao, Ding, Haoran, Tang, Matthew, Gandhi, Priyanka, Zhang, Zhan, and He, Zhe. Attention mechanism with bert for content annotation and categorization of pregnancy-related questions on a community q and a site. In *2020 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pp. 1077–1081, 2020. doi: 10.1109/BIBM49941.2020.9313379.
- Mead, A. Review of the development of multidimensional scaling methods. *Journal of the Royal Statistical Society. Series D (The Statistician)*, 41(1):27–39, 1992. ISSN 00390526, 14679884. URL <http://www.jstor.org/stable/2348634>.
- Pietiläinen, Aapo and Ji, Shaoxiong. AaltoNLP at SemEval-2022 task 11: Ensembling task-adaptive pretrained transformers for multilingual complex NER. In *Proceedings of the 16th International Workshop on Semantic Evaluation (SemEval-2022)*, pp. 1477–1482, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.semeval-1.202. URL <https://aclanthology.org/2022.semeval-1.202>.
- Ribeiro, Marco Tulio, Singh, Sameer, and Guestrin, Carlos. “why should i trust you?”: Explaining the predictions of any classifier, Feb 2016. URL <https://arxiv.org/abs/1602.04938>.
- Risch, Julian and Krestel, Ralf. Bagging BERT models for robust aggression identification. In *Proceedings of the Second Workshop on Trolling, Aggression and Cyberbullying*, pp. 55–61, Marseille, France, May 2020. European Language Resources Association (ELRA). ISBN 979-10-95546-56-6. URL <https://aclanthology.org/2020.trac-1.9>.
- Štrumbelj, Erik and Kononenko, Igor. Explaining prediction models and individual predictions with feature contributions. *Knowledge and Information Systems*, 41(3):647–665, Dec 2014. ISSN 0219-3116. doi: 10.1007/s10115-013-0679-x. URL <https://doi.org/10.1007/s10115-013-0679-x>.
- Sundararajan, Mukund, Taly, Ankur, and Yan, Qiqi. Axiomatic attribution for deep networks. Jun 2017.
- Utans, Joachim. Weight averaging for neural networks and local resampling schemes. In *Proc. AAAI-96 Workshop on Integrating Multiple Learned Models*. AAAI Press, pp. 133–138. Citeseer, 1996.
- Vaswani, Ashish, Shazeer, Noam, Parmar, Niki, Uszkoreit, Jakob, Jones, Llion, Gomez, Aidan N., Kaiser, Lukasz, and Polosukhin, Illia. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL <http://arxiv.org/abs/1706.03762>.
- Warstadt, Alex, Singh, Amanpreet, and Bowman, Samuel R. Neural network acceptability judgments. *arXiv preprint arXiv:1805.12471*, 2018.

8. Appendix

hyperparamter	value
model	bert-base-cased
	roberta-base
	prajjwal1/bert-medium
tokenizer	bert-base-cased
	roberta-base
	prajjwal1/bert-medium
epochs	100
batch size	4
early stopping	10
optimizer	adam
learning rate	1e-6
random seed	[0, 42, 121]

Table 4. Hyperparameteres for our BERTForSequenceClassifica-
tion or RoBERTaForSequenceClassificationmodel.