Hate Speech Detection: Ensemble Methods

Subjectivity Mining 2022

Ilia Markov

Faculty of Humanities
Computational Linguistics and Text Mining Lab (CLTL)
i.markov@vu.nl

Outline

- Ensemble methods: introduction
- Voting ensemble algorithms:
  - Hard majority voting
  - Soft majority voting
- Stacked generalization (stacking)
- Error analysis of ensemble output and recent trends

Content warning

This presentation contains examples of language which may be offensive to some of you. They do not represent the views of the lecturer.

Ensemble methods: introduction

- Ensemble methods: techniques that create multiple models and then combine them to produce improved results
- Popular in NLP and hate speech detection: most of the top-ranked teams in the recent hate speech shared tasks used ensemble methods

Ensemble methods: introduction

Three main families of ensemble learning algorithms:

- **Bagging ensembles**: involve training many high variance models on different samples of the training dataset (Random forest, Bagging classifier)
- **Boosting ensembles**: involve adding models sequentially to correct the predictions of prior models (Gradient boosting, AdaBoost)
- Stacking ensembles (our focus): involve using a model to learn how to best combine the predictions from models (voting, stacked generalization)

Bagging ensembles: key ideas

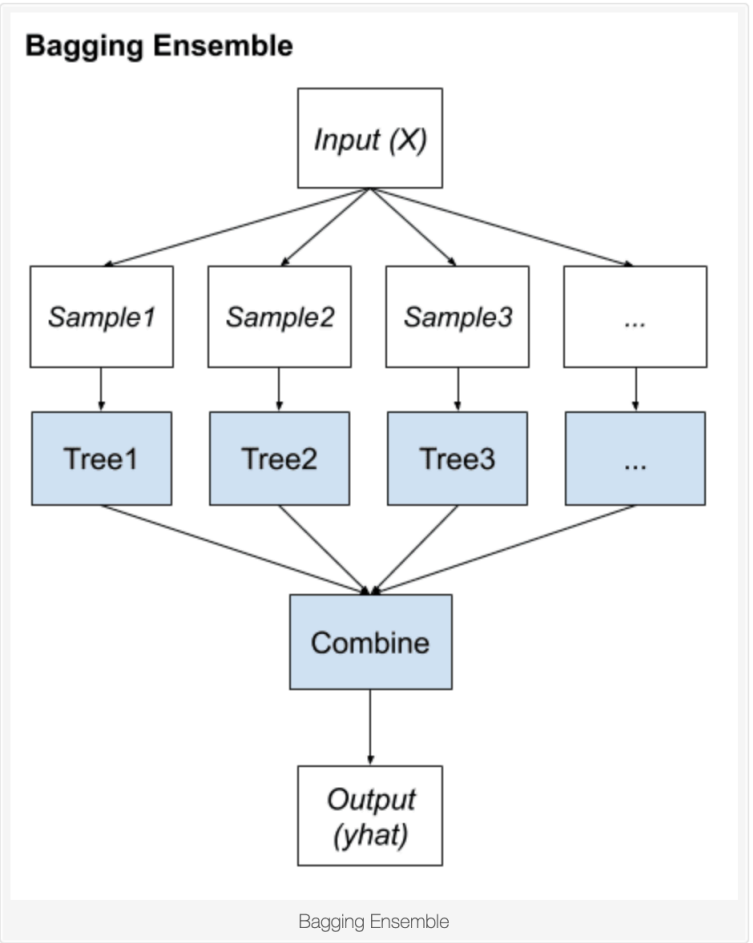
- **Bagging** (or bootstrap aggregation) is an ensemble learning method that seeks a diverse group of ensemble members by varying the training data.
- Typically involves using a single ML algorithm (unpruned decision tree), and training each model on a unique sample of the same training data (bootstrap sample).
- The predictions made by the ensemble members are then combined using simple statistics, e.g., voting or averaging.

Bagging ensembles: key elements

- Bootstrap samples of the training dataset
- Unpruned decision trees fit on each sample
- Simple voting or averaging of predictions

The contribution of bagging: varying of the training data used to fit each ensemble member, which results in skillful but different models.

Source: Jason Brownlee's tutorial



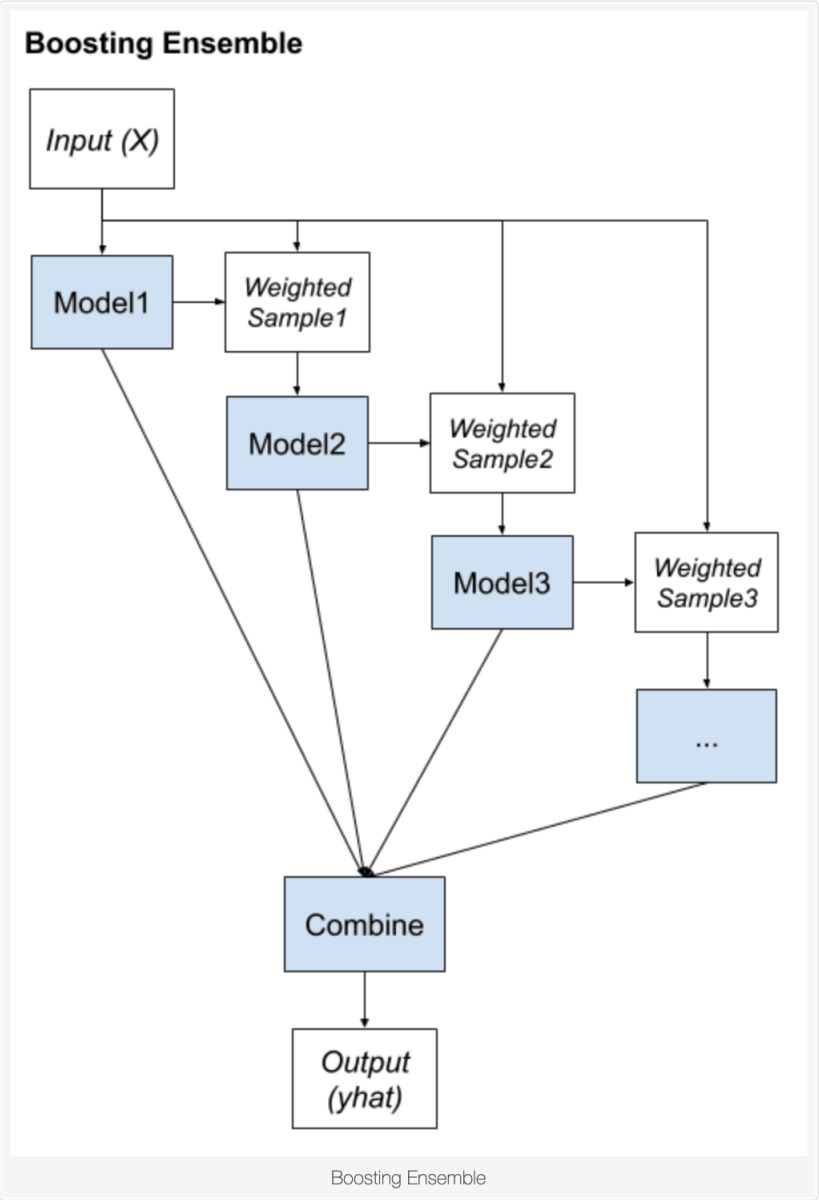
Bagging Ensemble

Boosting ensembles: key ideas

- **Boosting** is an ensemble method that seeks to change the training data to focus attention on examples that previous models have classified wrong.
- The key property of boosting ensembles is the idea of correcting prediction errors.
- The models are fit and added to the ensemble sequentially: the second model attempts to correct the predictions of the first model, the third corrects the second model, and so on.

Boosting ensembles: key elements

- Bias training data toward those examples that are hard to predict
- Iteratively add ensemble members to correct predictions of prior models
- Combine predictions using a weighted average of models



Boosting Ensemble

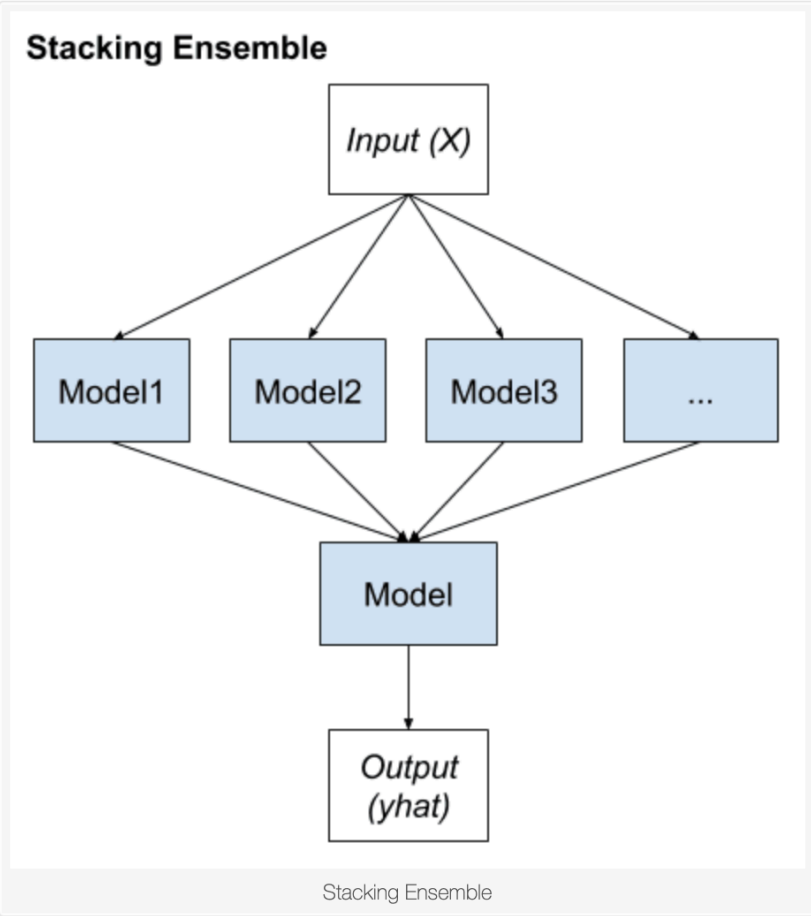
Stacking ensembles: key ideas

Stacking (stacked generalization) is an ensemble method that seeks a diverse group of members by varying the model types fit on the training data and using a model to combine predictions.

Stacking ensembles: key elements

- Unchanged training dataset
- Different ML/DL algorithms for each ensemble member
- ML/DL model to learn how to best combine predictions

Diversity comes from the different ML/DL models used as ensemble members: well-performing models with uncorrelated predictions



Stacking Ensemble

Hard majority-voting ensemble

Ensemble methods: hard majority voting

Hard majority-voting ensemble: selecting the label that is most often predicted (at least 3 classifiers)

Test set predictions:

Text	Model 1 (BERT)	Model 2 (RoBERTa)	Model 3 (SVM)	Hard voting
Text1...	1	0	1	1
Text2...	0	0	1	0

$$\hat{y} = mode\{C_1(\mathbf{x}), C_2(\mathbf{x}), \ldots, C_m(\mathbf{x})\}$$

predict the class label $\hat{y}$ via majority (plurality) voting of each classifier C_j; $mode$ - the most frequent number

Ensemble methods: performance

Ensemble performs well:

- Good performance of component models (base-models)
- Different errors by component models: uncorrelated predictions (Pearson's correlation coefficients)

Pearson correlation coefficient – a statistical measure of linear correlation between two sets of data. A value between -1 and 1; 1 $\rightarrow$ perfect correlation

Compute the correlation between two dataframe columns

Hard majority voting (Markov and Daelemans, 2021): models and datasets

Markov and Daelemans (2021): hard majority-voting ensemble of pre-trained language models and a conventional ML approach

Models:

- **BERT** (Devlin et al., 2019)
- **RoBERTa** (Liu et al., 2019)
- **SVM**: stylometric and emotion-based approach (Markov et al., 2021)

Datasets:

- **FRENK** (Ljubešić et al. 2019) (Facebook comments)
- **OLID** (Zampieri et al., 2019) (tweets)

Hard majority voting (Markov and Daelemans, 2021): results

Results (binary classification):

In-domain

	FRENK			OLID		
Model	Precision	Recall	F1	Precision	Recall	F1
BoW	71.0	70.8	70.9	75.9	70.9	72.5
CNN	76.8	76.6	76.7	81.8	77.8	79.4
LSTM	73.3	72.5	72.8	78.2	75.1	76.4
BERT	78.3	78.4	78.3	82.3	82.0	82.2
RoBERTa	78.4	78.7	78.5	80.2	79.7	80.0
SVM	77.8	76.4	77.0	82.3	76.1	78.3
Ensemble	**80.0**	**79.5**	**79.7***	**84.7**	**82.0**	**83.2**

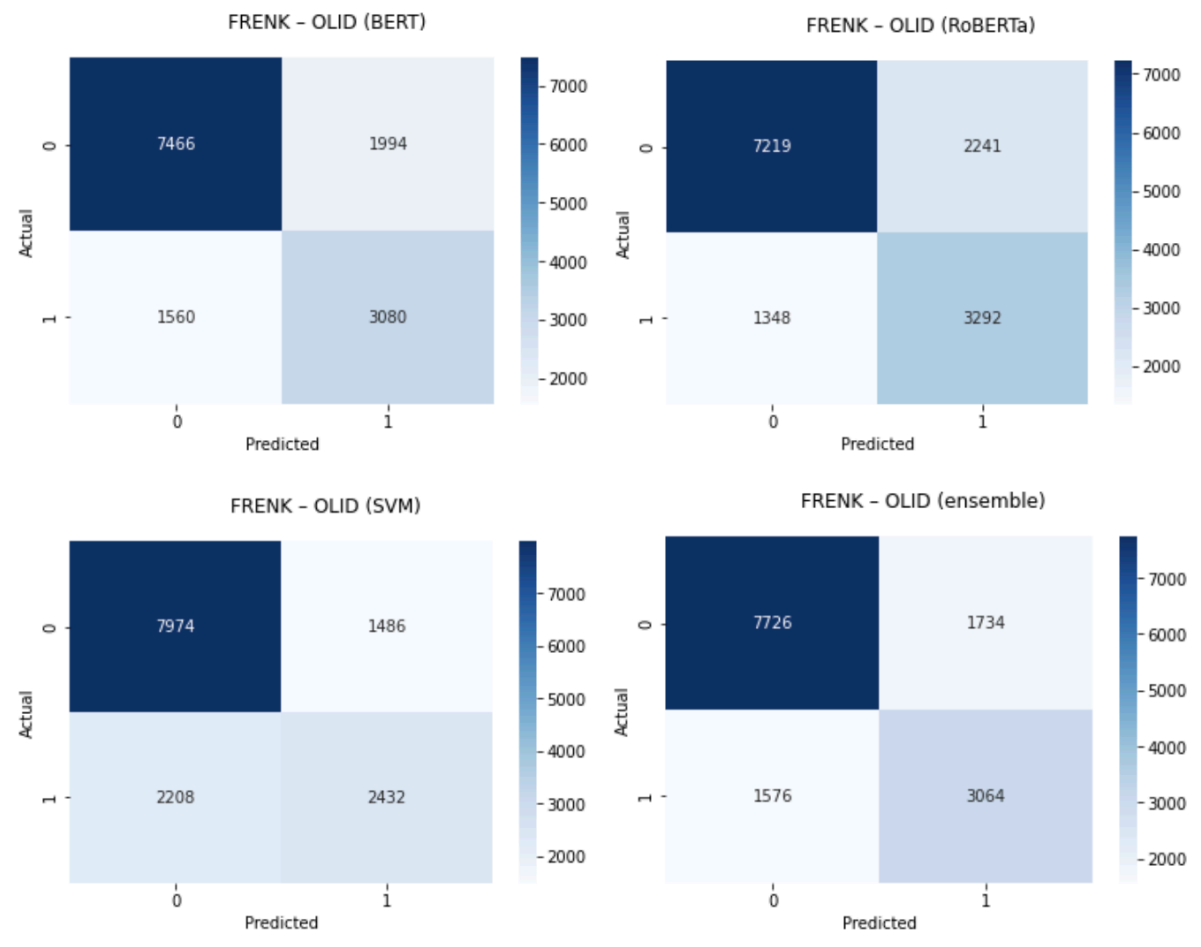
Cross-domain

	OLID – FRENK			FRENK – OLID		
Model	Precision	Recall	F1	Precision	Recall	F1
BoW	70.3	64.9	65.5	66.3	63.1	63.8
CNN	70.8	65.6	66.3	65.9	67.6	66.0
LSTM	68.0	66.1	66.6	67.5	65.9	66.5
BERT	70.5	68.8	69.4	71.7	72.7	72.1
RoBERTa	**73.9**	68.2	69.2	71.9	73.6	72.4
SVM	70.2	67.0	67.7	70.2	68.4	69.0
Ensemble	73.1	**68.8**	**69.7***	**73.5**	**73.9**	**73.6***

Ensemble outperforms all individual models by a significant margin

Uncorrelated predictions by pre-trained languages models and SVM

Hard majority voting (Markov and Daelemans, 2021): confusion matrices



Improvement due to the reduces false positive rates.

Soft majority-voting ensemble

Ensemble methods: soft majority voting

Soft majority-voting ensemble: selecting the label with the highest probability (at least 2 classifiers)

Test set prediction probabilities (negative class/positive class):

Text	Model 1 (BERT 1)	Model 2 (BERT 2)	...	Model 15 (BERT 15)	Soft voting
Text1...	0.2/0.8	0.3/0.7	...	0.4/0.6	1
Text2...	0.2/0.8	0.3/0.7	...	0.9/0.1	0

$$\hat{y} = \underset{j}{\text{argmax}} \sum_{i=1}^{n} p_{i,j}$$

where $p_{i,j}$ is the probability for class label j predicted by the i-th classifier (out of n classifiers).

Sums up the probability mass assigned per class label and chooses the label with the highest probability as the ensemble's prediction.

Risch and Krestel (2020)

Risch and Krestel (2020): ensemble of multiple fine-tuned BERT models

Models: **15 fine-tuned BERT models** trained on slightly different subsets of the data (varying training and validation splits, bootstrap aggregation or bagging) or with different random seeds (reason for improvement)

Dataset:

- Corpus of Misogyny and Aggression (comments on YouTube videos) (Bhattacharya et al., 2020)
- Used in the TRAC-2020 shared task: Evaluating Aggression Identification in Social Media (Kumar et al., 2020)

Risch and Krestel (2020): results

Results (3 classes: non-aggressive, covertly aggressive, overtly aggressive):

Best-performing system for five out of six subtasks in the TRAC-2020 shared task:

	English		Hindi		Bangla	
	Task A	Task B	Task A	Task B	Task A	Task B
Our Submission	**80.29**	85.14	**81.28**	**87.81**	**82.19**	**93.85**
Best Other Submission	75.92	**87.16**	79.44	86.89	80.83	92.97

Ensemble achieves a two percentage points higher F1-score than single models.

Voting algorithms: limitation

Limitation: all models contribute equally to the prediction (problem if some models are good in some situations and poor in others)

Solution:

- Use a weighted average or weighted voting of the contributing models (blending)
- Use a ML/DL model to learn when and how much to trust each model when making predictions (stacked generalization or stacking)

Stacked generalization (stacking)

Stacked generalization (stacking)

Question: given multiple ML/DL models that are skillful on a problem, but in different ways, how do you choose which model to use (trust)?

Solution: use another ML/DL model (a meta-learning algorithm, meta-model) that learns when to use or trust each model in the ensemble.

Stacked generalization (stacking)

The meta-model is trained on the predictions made by base models on out-of-sample data.

The most common approach to preparing the training dataset for the meta-model is via k-fold cross-validation of the base models.

K-fold cross-validation and test predictions

Training set predictions (k-fold cross-validation):

Text	Model 1 (BERT)	Model 2 (RoBERTa)	Model 3 (SVM)	...	Model n (CNNs)	True label
Text1...	1	0	1	...	1	1
Text2...	0	0	1	...	1	0
Text3...	1	1	1	...	1	1
Text4...	0	1	0	...	0	0

Test set predictions:

Text	Model 1 (BERT)	Model 2 (RoBERTa)	Model 3 (SVM)	...	Model n (CNNs)	Predicted label
Text...	1	0	1	...	1	1
Text...	0	0	1	...	1	0

K-fold cross-validation: example

5-fold cross-validation SVM output (example):

```
import pandas as pd
from sklearn.model_selection import StratifiedKFold
from sklearn.svm import LinearSVC

# 5FCV output for ensemble
rskf = StratifiedKFold(n_splits=5, shuffle=True)
pred = []
gold = []
index = []
for train_index, test_index in rskf.split(X_train, Y_train):
    clf_svc = LinearSVC(random_state=0)
    clf_svc.fit(X_train[train_index], Y_train[train_index])
    Y_pred_10FCV = clf_svc.predict(X_train[test_index])

    gold.extend(Y_train[test_index])
    pred.extend(Y_pred_10FCV)
    index.extend(test_index)

output = pd.DataFrame(columns = ['label','predicted','ind'])
output.label, output.predicted, output.ind = gold, pred, index
output['id'] = [train.id.to_list()[idx] for idx in index]
output = output[['id', 'label', 'predicted']]
```

K-fold cross-validation: example

5-fold cross-validation BERT output (example):

```
import pandas as pd
from sklearn.model_selection import StratifiedKFold
from simpletransformers.classification import ClassificationModel

# 5FCV output for ensemble
rskf = StratifiedKFold(n_splits=5, shuffle=True)
pred = []
probabilities = []
gold = []
index = []
for train_index, test_index in rskf.split(train['text'], train['labels']):
    train_df = train.iloc[train_index]
    test_df = train.iloc[test_index]
    model = ClassificationModel('bert', 'bert-base-cased', args={'reprocess_input_data': True,
                                                                 'overwrite_output_dir': True},
                                                                  use_cuda=True)

    model.train_model(train_df)
    predictions, prob = model.predict(test_df.text.to_list())

    gold.extend(test_df['labels'])
    pred.extend(predictions)
    probabilities.extend(softmax(prob, axis=1)[:,1])
    index.extend(test_index)

output = pd.DataFrame(columns = ['label', 'probabilities', 'predicted', 'ind'])
output.label, output.probabilities, output.predicted, output.ind = gold, probabilities, pred, index
output['id'] = [train.id.to_list()[idx] for idx in index]
output = output[['id', 'label', 'probabilities', 'predicted']]
```

Tutorial: K-Fold as Cross-Validation with a BERT Text-Classification Example

Additional information

The training data for the meta-model may also include the inputs to the base models, e.g., input elements of the training data.

This can provide an additional context to the meta-model as to how to best combine the predictions.

Additional features

Additional features to represent input information: e.g., message length in terms of tokens, message length in terms of characters

Training set predictions (k-fold cross-validation):

Text	Model 1 (BERT)	Model 2 (RoBERTa)	Model 3 (SVM)	...	Model n (CNNs)	Additional feature 1	Additional feature 2	True label
Text1...	1	0	1	...	1	6	4	1
Text2...	0	0	1	...	1	2	1	0
Text3...	1	1	1	...	1	0	1	1
Text4...	0	1	0	...	0	0	1	0

Test set predictions:

Text	Model 1 (BERT)	Model 2 (RoBERTa)	Model 3 (SVM)	...	Model n (CNNs)	Additional feature 1	Additional feature 2	Predicted label
Text...	1	0	1	...	1	4	3	1
Text...	0	0	1	...	1	0	1	0

Risch and Krestel (2018): models

Risch and Krestel (2018): ensemble of a neural network and 3 logistic regression classifiers (First Shared Task on Aggression Identification, TRAC-2018)

Shared task data: aggression-annotated Facebook posts; 3 classes: overtly aggressive (OAG), covertly aggressive (CAG), non-aggressive (NAG)

Models:

- **Recurrent neural network (RNN):** bi-directional GRU (extract orderly information, interpret a document as a sequence of words or characters)
- **3 logistic regression classifiers**
  - Character n-grams (n = 2–6), tf-idf
  - Word n-grams (n = 1–2), tf-idf
  - Syntactic features: emoticons, the number of words, the proportion of uppercase characters to lowercase characters, the number of negation words, the polarity of the post (VADER)

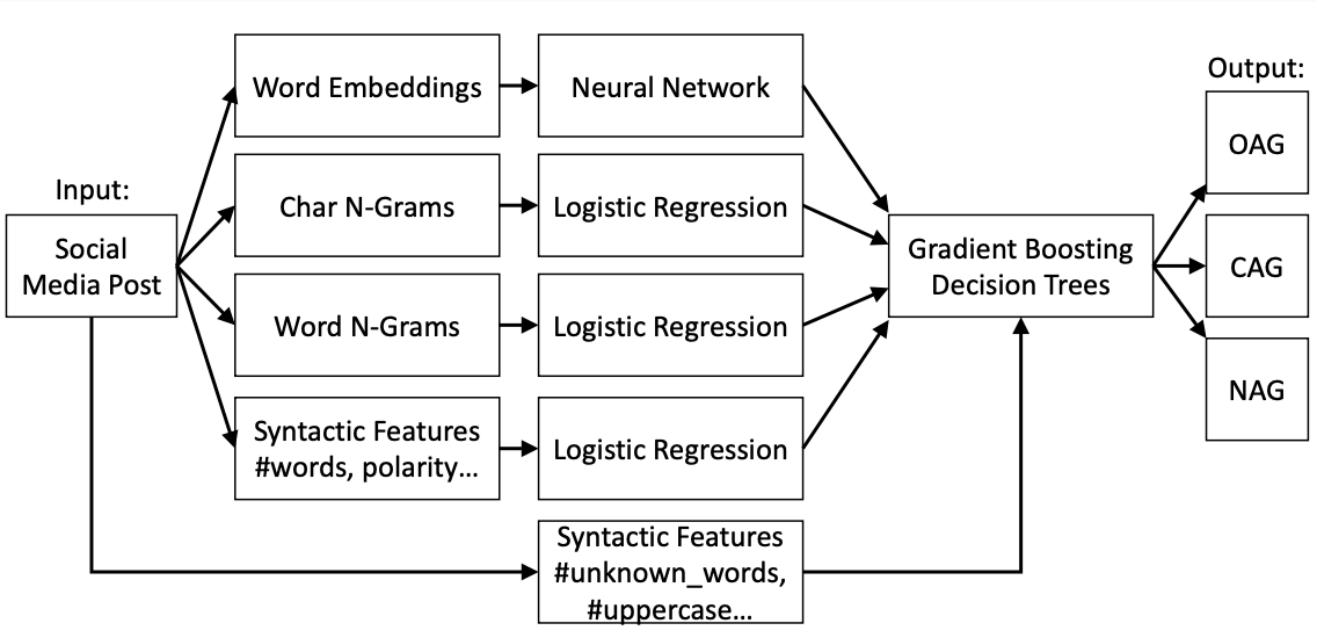
Risch and Krestel (2018): additional features

10-fold cross-validation (produce predictions for each message in the training + test data) + additional features to represent input information:

- Length (number of characters)
- Relative number of uppercase characters
- Relative number of non-alpha characters
- Relative number of exclamation marks
- Relative number of how many words in the comment have a GloVe embedding (to measure how many uncommon words are used in the comment)

Gradient boosting classifier (Friedman, 2001) as a meta-model

Risch and Krestel (2018): ensemble architecture



Source: Risch and Krestel (2018)

Risch and Krestel (2018): results

Results for English and Hindi (weighted F1-score):

System	F1 En FB	F1 En FB augm.	F1 Hi FB
Random Baseline	0.3324	0.3395	0.3425
RNN	0.5722	0.5846	0.5413
Word N-Grams	0.5764	0.5766	0.5883
Char N-Grams	0.5803	0.5791	0.6103
Feature Selection	0.3966	0.3871	0.3701
Ensemble	**0.6060**	**0.6084**	**0.6292**

Ensemble outperforms all individual models

RNN and logistic regression – low correlation

van Aken et al. (2018): models

van Aken et al. (2018): ensemble of deep learning and shallow approaches

Models:

- **Logistic regression** (word n-grams and character n-grams)
- **4 recurrent neural networks (RNNs)** approaches:
  - **LSTMs**
  - **BiLSTMs** (compensate certain errors on long range dependencies)
  - **Bidirectional GRU** (Gated Recurrent Unit)
  - **Bidirectional GRU with attention layer** ("attention mechanisms are suitable for identifying specific small regions indicating hatefulness in long comments")
- **CNNs**

van Aken et al. (2018): ensemble strategy

5-fold cross-validation + gradient boosting classifier (Friedman, 2001)

An ensemble deciding which of the single classifiers is most powerful on a specific kind of comment.

van Aken et al. (2018): datasets

Datasets:

- Comments on Wikipedia talk pages: toxic comment classification challenge (6 classes, multi-label classification)
- Twitter dataset (Davidson et al., 2017), hate speech detection (3 classes, multi-class classification)

van Aken et al. (2018): results

Model	Wikipedia				Twitter			
	P	R	F1	AUC	P	R	F1	AUC
CNN (FastText)	.73	.86	.776	.981	.73	.83	.775	.948
CNN (Glove)	.70	.85	.748	.979	.72	.82	.769	.945
LSTM (FastText)	.71	.85	.752	.978	.73	.83	.778	.955
LSTM (Glove)	**.74**	.84	.777	.980	.74	.82	.781	.953
Bidirectional LSTM (FastText)	.71	.86	.755	.979	.72	.84	.775	.954
Bidirectional LSTM (Glove)	**.74**	.84	.777	.981	.73	**.85**	.783	.953
Bidirectional GRU (FastText)	.72	.86	.765	.981	.72	.83	.773	.955
Bidirectional GRU (Glove)	.73	.85	.772	.981	.76	.81	.784	.955
Bidirectional GRU Attention (FastText)	**.74**	**.87**	**.783**	**.983**	.74	.83	**.791**	**.958**
Bidirectional GRU Attention (Glove)	.73	**.87**	.779	**.983**	**.77**	.82	**.790**	.952
Logistic Regression (char-ngrams)	**.74**	.84	.776	.975	.73	.81	.764	.937
Logistic Regression (word-ngrams)	.70	.83	.747	.962	.71	.80	.746	.933
Ensemble	**.74**	**.88**	**.791**	**.983**	.76	.83	**.793**	.953

- Ensemble outperforms all individual models
- Combining shallow learner approach with neural nets is highly effective (uncorrelated predictions)

Markov et al. (2022): models

Markov et al. (2022): ensemble of pre-trained language models and conventional ML approaches for Dutch

Models for Dutch:

- **BERTje** (de Vries et al., 2019): pre-trained using the same architecture and parameters as the original BERT model on a dataset of 2.4 billion tokens.
- **RobBERT** (Delobelle et al., 2020): pre-trained using the RoBERTa training regime on a corpus of 6.6 billion words
- **SVM**: stylometric and emotion-based approach (Markov et al., 2021)

Stratified 5-fold cross-validation

Markov et al. (2022): features

Additional features to represent input information (feature engineering):

- Number of emotion-conveying words in a message from the Dutch LiLaH emotion lexicon (Ljubešić et al. 2020)
- Number of hateful words from the POW hate speech lexicon (De Smedt et al., 2020)
- Number of first-person personal pronouns, which are often used as features for hate speech detection to distinguish between 'us' and 'them'
- Message length in terms of words
- Message length in terms of characters

Gradient boosting classifier (Friedman, 2001); scikit-learn implementation

Markov et al. (2022): datasets

Datasets: two recent Dutch social media datasets

- LiLaH (Facebook comments); LGBT and migrants topics
- **DALC**: The Dutch Abusive Language Corpus (Caselli et al., 2021) (tweets); variety of topics related to events occurred between 2015 and 2020

Markov et al. (2022): results

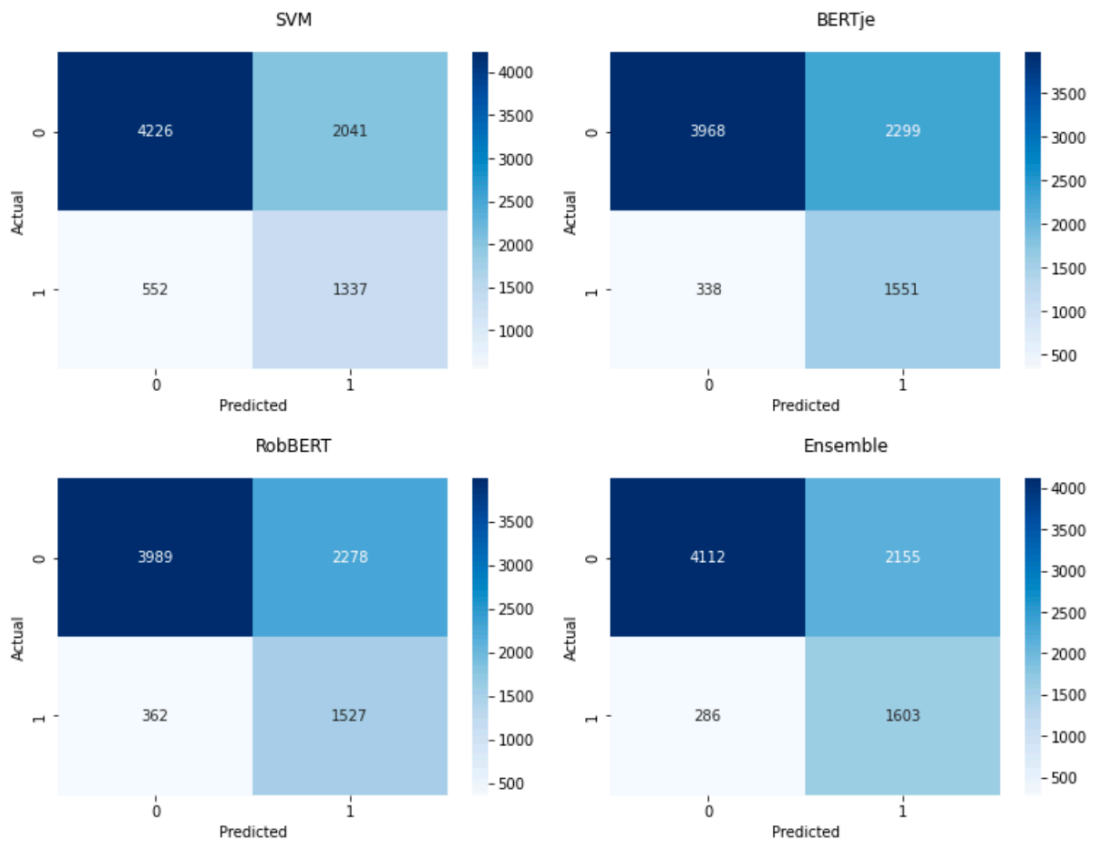
Results (binary classification):

In-domain						
Model	**LiLaH**			**DALC**		
	Precision	Recall	F1	Precision	Recall	F1
Majority baseline	28.0	50.0	35.9	33.2	50.0	39.9
SVM	75.3	75.3	75.3	81.0	69.5	71.3
BERTje	77.6	77.1	77.3	82.5	72.7	74.7
RobBERT	77.4	75.7	76.0	82.7	73.3	75.3
Ensemble	**78.8**	**78.2**	**78.4***	**84.9**	**75.0**	**77.2***
Cross-domain						
Model	**Train DALC, test LiLaH (test)**			**Train LiLaH, test DALC (test)**		
	Precision	Recall	F1	Precision	Recall	F1
SVM	70.7	59.4	55.5	67.7	69.4	67.9
BERTje	77.0	62.1	58.6	70.0	72.2	69.9
RobBERT	75.2	60.4	56.2	70.2	72.3	70.4
Ensemble	**76.3**	**62.5**	**59.3**	**74.0**	**76.8**	**74.0***

Ensemble significantly outperforms all the individual models both in the in-domain and cross-domain hate speech detection settings

Markov et al. (2022): confusion matrices

Confusion matrices: training on LiLaH, testing on DALC



Ensemble performs better than the individual models in terms of both false positives (except for SVM) and false negatives

Base-models and meta-model

- **Base-models** are often complex and diverse: deep learning / robust machine learning approaches (good performance, uncorrelated predictions)
- **Meta-model** is often simple:
  - Gradient boosting
  - AdaBoost
  - Decision trees
  - Random forest
  - SVM
  - Logistic regression
  - Neural network

Ensemble methods

‘+’ State-of-the-art performance: allow to improve upon best-performing individual model

‘+’ Give more stable predictions than a single model

‘-’ More complex than using a single ML/DL model

‘-’ Requires more computational resources

‘-’ Good performance of base-models

‘-’ Uncorrelated predictions

‘–’ Explainability: why a specific prediction was made

Error Analysis of Ensemble Output

Error analysis of ensemble results: common errors

van Aken et al. (2018) and Markov et al. (2022): error analysis of the ensembles output to identify common errors that most of the state-of-the-art approaches share (English and Dutch, respectively)

- **van Aken et al. (2018):** Wikipedia Talk Pages and Twitter datasets; neural networks (CNNs, LSTMs, BiLSTMs, ...) and ML (Logistic Regression) approaches
- **Markov et al. (2022):** DALC dataset (Twitter); DL (BERTje, RobBERT) and ML (SVM) approaches, additional features

Common errors across different ensemble strategies, platforms and languages

Error classes of false negatives: doubtful labels

False negatives (hateful messages that were missed by the algorithm):

- **Doubtful labels:** question the original label w.r.t the class definition (cf. subjectivity of the hate speech phenomenon)
  - Toxic or hateful content that is cited by the comment's author
  - Use of potentially toxic words within an explanation or self reproach

Example: *"No matter how upset you may be there is never a reason to refer to another editor as 'an idiot' "*

23% Wikipedia dataset, 9% Twitter dataset, 40% DALC dataset (largest error class)

$\Rightarrow$ Detailed annotation guidelines

$\Rightarrow$ Trained annotators

$\Rightarrow$ Others

Error classes of false negatives: toxicity without swear words

- **Toxicity without swear words:** hate speech may not contain hate or swear words at all

Example: *"she looks like a horse"*

50% Wikipedia dataset (largest error class), 18% Twitter dataset, 12% DALC dataset

$\Rightarrow$ Encoding context

$\Rightarrow$ Others

Error classes of false negatives: rhetorical questions

- **Rhetorical questions:**

Example: *“have you no brain?!?!”*

21% Wikipedia dataset, 10% Twitter dataset

⇒ Encoding context

⇒ Existence of question words and question marks

⇒ Others

Error classes of false negatives: metaphors and comparisons

- **Metaphors and comparisons:** figurative language is a chellenge in NLP and hate speech detection

Example: *"Who are you a sockpuppet for?"*

16% Wikipedia dataset, 14% DALC dataset

⇒ Encoding additional world knowledge to understand the meaning (retrieving external knowledge)

Example: *"Welcoming migrants is like going to the dentist."*

⇒ Others

Error classes of false negatives: idiosyncratic and rare words

- **Idiosyncratic and rare words:** misspellings, neologisms, obfuscations, abbreviations, slang words, swearing in another language

Example: *"fucc nicca yu pose to be pullin up"*

30% Wikipedia dataset, 43% Twitter dataset, 23% DALC dataset

Reflects the common language on Twitter

⇒ Train word embeddings on larger corpora with even more variant language

⇒ Character-level approaches

⇒ Others

Error classes of false negatives: sarcasm and irony

- **Sarcasm and irony:** texts usually state the opposite of what is really meant

Example: *“hope you’re proud of yourself. Another milestone in idiocy”*

11% Wikipedia dataset, 3% DALC dataset

⇒ Diverse training datasets with linguistic variation

⇒ Existence of laughing emojis and quotation marks

⇒ Others

Error classes of false positives: doubtful labels

False positives (non-hateful messages erroneously classified as hateful):

- **Doubtful labels**

Example: *"IF YOU LOOK THIS UP UR A DUMB RUSSIAN"*

53% Wikipedia dataset (one of the largest error classes), 10% Twitter dataset, 17% DALC dataset

⇒ Detailed annotation guidelines

⇒ Trained annotators

⇒ Others

Error classes of false positives: usage of swear words

- **Usage of swear words in false positives:** classifiers often learn that swear words are strong indicators for toxicity in comments, problematic when non-toxic comments contain such terms

Example: *"Oh, I feel like such an asshole now. Sorry, bud."*

60% Wikipedia dataset (largest error class), 77% Twitter dataset (largest error class), 7% DALC dataset

$\Rightarrow$ Diverse training datasets with linguistic variation

$\Rightarrow$ Others

Error classes of false positives: quotations or references

- **Quotations or references:** references to toxic or hateful language in actual non-hateful comments

Example: *"I deleted the Jews are dumb comment"*

17% Wikipedia dataset, 8% Twitter dataset

$\Rightarrow$ Encoding context

$\Rightarrow$ Others

Error classes of false positives: idiosyncratic and rare words

- **Idiosyncratic and rare words:** classifier misinterprets their meaning or slang that is often used in toxic language

Example: *"WTF man. Dan Whyte is Scottish"*

8% Wikipedia dataset, 7% Twitter dataset

⇒ Train word embeddings on larger corpora with even more variant language

⇒ Character-level approaches

⇒ Others

Error classes of false positives: counter narratives

- **Counter narratives:** non-aggressive response using credible evidence, factual arguments, alternative viewpoints; effective strategy to combat online hate speech.

Example: *"Isn't it unbelievable that Muslims are being blamed for the actions of ISIS who are slaughtering Muslims in Syria and Iraq? How stupid?!"*

10% DALC dataset

$\Rightarrow$ Encoding context

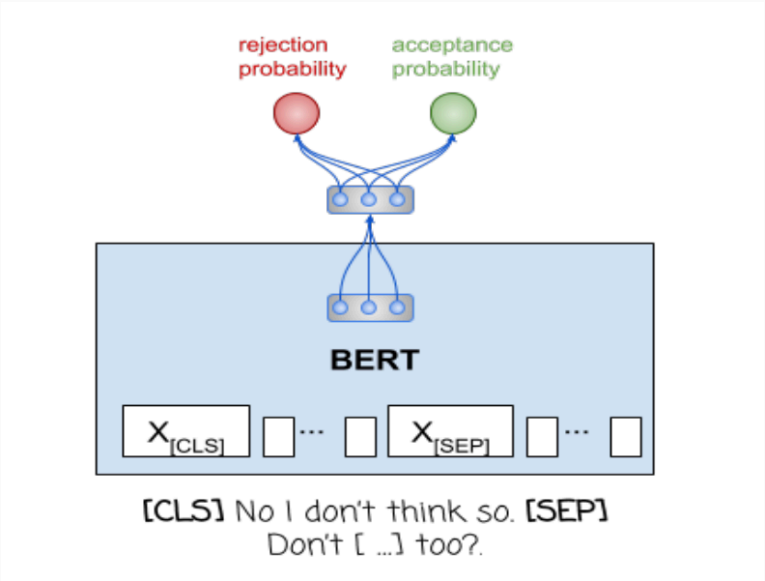
$\Rightarrow$ Others

Recent Trends

Recent strategies: encoding conversational context

Important for hate speech detection: “Go back home”

The text of the post / previous (‘parent’) comment in the discussion thread as context, e.g.:



Wikipedia conversations dataset. Concatenate the text of the parent and target comments, separated by BERT’s [SEP] token. Source: Pavlopoulos et al. (2020)

Context significantly affects annotation process; no evidence that leads to a large or consistent improvement in model performance

Recent strategies: encoding conversational context

Detect the target of hates speech (migrants or not). Manually annotate relevant conversational context in Dutch Facebook discussion threads:

- If context influences the annotators' decision to assign a label or the target of hate speech is not sufficiently clear without contextual information → indicate ID of comment/post that serves as context
- Concatenate using [SEP] token before fine-tuning BERTje

	Precision	Recall	F1-score
Comment (baseline)	0.65 (±0.008)	0.66 (±0.008)	0.63 (±0.011)
Comment & post	0.66 (±0.008)	0.67 (±0.004)	0.65 (±0.008)
Comment & preceding comment	0.64 (±0.007)	0.65 (±0.004)	0.63 (±0.015)
Comment & preceding comment & post	0.64 (±0.004)	0.65 (±0.008)	0.63 (±0.000)
Comment & context	**0.70** (±0.008)	**0.71** (±0.008)	**0.69** (±0.008)

Source: Markov et al. (2022, submitted)

Next step: identify relevant context automatically

Recent strategies: figurative language (hateful metaphors)

Lemmens et al. (2021): hateful metaphors as features for detecting the type and target of hate speech

Conceptual metaphor theory (Lakoff and Johnson, 1980): metaphors are utterances that describe a target concept in terms of a source concept: "he *attacked* my arguments", "I *destroyed* him during our discussion": conceptual metaphor *argument is war*

- Annotated Facebook dataset for metaphorical tokens and source domains (animals, dirt, body parts, diseases): ≈15% of messages contain metaphors; 20 source domains
- Used for SVM (1-2-grams, tf-idf) and BERT:
  "You're a *pig*" → You're a MET *pig* MET ANIMALS
  Improvement up to 4 F1 points

Next step: identify hateful metaphors automatically

Questions?

Assignment 4 (final)