

Hate Speech Detection: Methods

Subjectivity Mining 2022

Ilia Markov

Faculty of Humanities

Computational Linguistics and Text Mining Lab (CLTL)

i.markov@vu.nl

- Methods: introduction
- Lexicon-based approaches
- Conventional machine learning approaches
- Neural networks
- Transformer-based pre-trained language models

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.

Hate speech detection is an inherently challenging task that has recently received much attention from the natural language processing (NLP) community.

Discussion of automatic hate speech detection goes back approximately two decades (Spertus, 1997).

Recently renewed interest: new approaches, datasets with various annotation layers, shared tasks.

Performance increased significantly:

- Large amount of available data (social media platforms, niche platforms)
- Deep learning approaches
- Ensemble methods (week 5)

Challenges

Not a solved problem. Recent challenges include:

- Subjectivity of the hate speech phenomenon (low inter-annotator agreement)
- Figurative language (sarcasm, irony, metaphors)
- Encoding contextual information
- Cross-domain scalability: drop in performance when the training and test corpora are disjoint (topic, platform, community context)

These challenges constrain the performance and generalizability of hate speech detection models.

Shared tasks to speed up the development of the solutions.

- **Toxic comment classification challenge (2017)** (Wikipedia comments, English): different types of toxicity: toxic, severe toxic, insult, threat, obscene, identity hate
- **HatEval (2019)** (Twitter, English and Spanish): hate speech against immigrants and women (hate speech or not), aggressive behavior (aggressive or not), target (individual, group)
- **OffensEval (2019, 2020)** (Twitter, multiple languages): type (offensive or not) and target (targeted or untargeted; what is the target: individual, group, other) of offensive content (2020: 145 teams, all-time record)

- **Toxic Spans Detection (2021)** (comments to online articles, English): predict the spans of toxic posts that were responsible for the toxic label of the posts
- **EXIST (2022)** sEXism Identification in Social neTworks (Twitter, English and Spanish): sexist or not sexist; type of sexism (ideological and inequality, stereotyping and dominance, objectification, sexual violence, misogyny and non-sexual violence)
- **Diverse set of languages:** Spanish, Italian, Hindi, German, Arabic, Danish, Greek, Turkish, ...

Hate speech detection as a classification problem

Binary classification: the task of classifying textual content into hate or non-hate speech classes (the most common framing of the task)

Multi-class classification: each sample is labeled with exactly one class out of multiple classes, e.g., fine-grained types (e.g., threat, offense, violence) or the severity of hate speech

Multi-label classification: comments fulfilling different predefined criteria at the same time (e.g., toxic, severe toxic, insult, threat, obscene, identity hate)

Evaluation metrics

- **Precision:** of all messages predicted as hateful, how many are actually hateful
- **Recall:** how many of actually hateful messages our model predicted as hateful
- **F1-score:** harmonic mean of precision and recall;
macro-averaged: arithmetic mean of all the per-class F1-scores

Precision, recall and F1-score for each class

Area under the curve (AUC): way to summarize a precision-recall curve

Lexicon-Based Approaches

Existing lexicons: HurtLex

Lexicon-based approaches rely on information from a pre-prepared lexicon.

A large variety of existing lexicons for hate speech detection in multiple languages.

HurtLex ([Bassignana et al., 2018](#)): A Multilingual Lexicon of Words to Hurt

- Developed by a linguist (for Italian)
- Enriched with information from lexical resources (e.g., POS tags from MultiWordNet)
- Semi-automatically translated into more than 50 languages
- 8,288 entries (for EN, version 1.2)
- Download: <https://github.com/valeribasile/hurtlex>

Existing lexicons: HurtLex

The words (lemmas) are divided into 17 categories:

Label	Description
PS	negative stereotypes ethnic slurs
RCI	locations and demonyms
PA	professions and occupations
DDF	physical disabilities and diversity
DDP	cognitive disabilities and diversity
DMC	moral and behavioral defects
IS	words related to social and economic disadvantage
OR	plants
AN	animals
ASM	male genitalia
ASF	female genitalia
PR:	words related to prostitution
OM:	words related to homosexuality
QAS	with potential negative connotations
CDS	derogatory words
RE	felonies and words related to crime and immoral behavior
SVP	words related to the seven deadly sins of the Christian tradition

Existing lexicons: HurtLex

Macro-category indicating whether there is stereotype involved.

Hurtlex has a 2-level structure. Lemmas belong to one of these levels:

- Conservative: obtained by translating offensive senses of the words in the original lexicon.
- Inclusive: obtained by translating all the potentially relevant senses of the words in the original lexicon.

Good results for the misogyny identification shared task (rank 1 for one of the sub-tasks). Example HurtLex (csv format):

hurtlex_EN					
id	pos	category	stereotype	lemma	level
EN1382	n	qas	no	gag reel	inclusive
EN7077	a	cds	no	snotty	conservative
EN6856	n	is	yes	mendicant	conservative
EN5485	n	re	no	maffias	conservative
EN5024	n	cds	no	lying in trade	conservative
EN6950	n	re	no	yeargh	inclusive












Existing lexicons: POW

Profanity and offensive words (POW): Multilingual fine-grained lexicons for hate speech ([De Smedt et al., 2020](#))

- Constantly growing
- 4chan and 8chan embeddings + collected manually (for EN)
- 5 languages: English, German, French, Dutch, Hungarian
- Available upon request

POW: categories and toxicity scores









Words and word combinations are grouped into 11 categories:

HATE		Words that relate to negativity (e.g., <i>lame, worthless</i>), anger (<i>disgusting, hate, kick, rage</i>), cynicism (<i>we're doomed</i>) and sarcasm (<i>"very fine people"</i>).
SHIT		Words that relate to profanity (e.g., <i>damn, piss, shit</i>), in particular vulgar name-calling (<i>damn sambo, pisslam = piss + Islam</i>) and swearing (<i>BS, WTF</i>).
FUCK		Words that relate to pornography . (e.g., <i>cunt, dick, fuck</i>), in particular with regard to sex crimes (<i>goat fucker, pedo, rapist</i>).
FOOL		Words that relate to ridicule (e.g., <i>deplorable, poor snowflake, tinfoil hat</i>), in particular insults of intelligence (<i>degenerate, dotard, retard</i>).
SCUM		Words that relate to dehumanization (e.g., <i>cum dumpster, rat, scum, thug, vermin</i>) or defamation (<i>fake news peddler, treasonous dog</i>).
SLUT		Words that relate to sexism (e.g., <i>gay, lesbian</i>), on the basis of sexual orientation (<i>fag</i>), sexuality (<i>slut</i>), gender (<i>bitch</i>) and gender stereotypes (<i>coward, cuck</i>).
GOOK		Words that relate to racism (e.g., <i>black bitch, white trash</i>), on the basis of race (<i>nigger</i>), ethnicity (<i>hebrezz</i>), nationality (<i>africoon, chexican</i>) and looks (<i>fatso</i>).
HELL		Words that relate to religious ideology (e.g., <i>Christians, Jews, Muslims</i>), in particular islamophobia (<i>hatebeard</i>), jihadism (<i>infidel</i>) and antisemitism (<i>lolocaust</i>).
HEIL		Words that relate to political ideology (e.g., <i>communist, fascist, traitor</i>), in particular activism (<i>Antifa, Pegida</i>), extremism (<i>Islamic State</i>) and propaganda (<i>Infowars</i>).
PLOT		Words that relate to conspiracy (e.g., <i>fake news, hoax</i>), including government cover-up (<i>deep state, NWO</i>), doomsday (<i>lab virus</i>) and the occult (<i>Thule Society</i>).
KILL		Words that relate to conflict (e.g., <i>civil war, riot, terror</i>), including violence (<i>kill, shoot</i>), threats (<i>kill you, shoot you</i>) and extortion (<i>dig up dirt</i>).

Existing lexicons: POW

Each word or word combination has a toxicity score (0–4): 0 (neutral), 1 (tendentious), 2 (demeaning), 3 (offensive; low, biased, vulgar) or 4 (extremely offensive).

The scores and categories were assigned manually by multiple, diverse experts in linguistics, social and political sciences, and security.

SCORE	WORD(S)								
★★★★	<i>black bitch</i>	○	●	○	●	●	○	○	○
★★★★☆	<i>kill crusaders</i>	○	○	○	○	○	●	●	●
★★★☆☆	<i>disgusting</i>	●	●	○	○	○	○	○	○
★★☆☆☆	<i>tinfoil hat</i>	○	○	●	○	○	○	●	○

black b*** = dehumanization, sexism, racism; toxicity score = 4 (extremely offensive)

Ways to construct lexicons

- Manual construction: experts, crowd
- Semi-automatic construction: extension and enrichment (e.g., synonyms from WordNet)
- Automatic construction ([Zhu et al., 2021](#)):
 - Lexicon is mined from training data by a statistical strategy:
 - $\text{toxic_score}(w) = \#w_{\text{in_toxic_span}} / \#w_{\text{in_whole_corpus}}$,
 - $\#w_{\text{in_toxic_span}}$ - count of appearances of word w in toxic spans
 - $\#w_{\text{in_whole_corpus}}$ - count of appearances of word w in the whole corpus
 - words with a toxic score greater than a given threshold are selected
 - Expansion with WordNet: collect synsets of each toxic from WordNet
 - Expansion with GloVe: collect the nearest similar words by calculating cosine similarity of GloVe vectors

Zhu et al. (2021): results

Results for toxic span detection.

Spans: sequence of words that attribute to the post's toxicity

Offsets	Post
{9, ..., 13}	You're an idiot .

	# of words	P(%)	R(%)	F1(%)
Ensemble	-	75.01	89.66	70.83
Lexicon ¹ (Wiegand et al., 2018)	551	75.13	44.47	33.07
Lexicon ² (Wiegand et al., 2018)	2989	66.22	72.01	50.98
Lexicon ^{original} (Our)	119	76.71	82.22	64.98
Lexicon ^{wordnet} (Our)	231	72.56	84.05	64.09
Lexicon ^{glove} (Our)	186	73.98	83.34	64.19

Expanded lexicons:

- Improved recall; decreased precision
- There are non-toxic words in the synonyms found through WordNet / GloVe

Toxic spans detection

Toxic Spans Detection shared task (2021)

Offsets	Post
{9, ..., 13}	You're an idiot .

- Lexicon-based: list of toxic words for lookup operations
- Word-level BIO tags
- Model-specific or model-agnostic rationale extraction mechanisms to produce toxic spans as explanations of the decisions of the classifier

Encoding lexicon information: lexicon-lookup approaches

Given a reference dictionary/lexicon of profanities, abusive terms, slurs, etc., if a message contains one or more of the terms in the dictionary, then it is labeled as hateful (Caselli et al., 2021).

- Language: Dutch
- Lexicon: 847 potentially abusive terms from HurtLex v1.2 (Bassignana et al., 2018) + list with 256 culturally specific terms (e.g., names of diseases)
- Results (binary classification):

System	Class	Precision	Recall	Macro-F1
MFC	ABU	0	0	0.399
	NOT	0.664	1.0	
Dictionary	ABU	0.716	0.433	0.685
	NOT	0.761	0.913	
SVM	ABU	0.858	0.323	0.655
	NOT	0.740	0.973	
BERTje	ABU	0.850	0.500	0.748
	NOT	0.791	0.955	

- Used as a baseline; by organizations

Encoding lexicon information: supervised classification

- Machine learning: abusive terms from a lexicon are used as (additional) features for classifiers ([Markov et al., 2022](#))
- Deep learning: combine BERT with features extracted from a hate speech lexicon ([Koufakou et al., 2020](#))

Lexicon-based approaches: summary

Various existing lexicons and ways to construct lexicons and exploit lexicon information.

Lexicon-based approaches are often used in industry, as a baseline (lexicon-lookup) or as external information for supervised classification.

- ‘+’ Variety of available lexicons in different languages; multilingual resources
- ‘+’ Explainability (used by many organizations)
- ‘-’ Implicit hate speech: hate speech that is not explicitly expressed by means of profanities, slurs or insults
- ‘-’ Offensive words in a non-hateful content
- ‘-’ Hate speech lexicons become outdated quickly*

*Only some of the advantages and disadvantages for each approach are presented

Conventional Machine Learning Approaches

Commonly-used features

Conventional ML approaches: manually selected features are combined into input vectors and directly used for classification

- **Words (bag-of-words):** lexical choices
- **Word n-grams:** lexical choices, context around each word (larger and sparser feature set)
- **Character n-grams:** lexical and syntactic information, punctuation and capitalization information, spelling variations
- **Their combinations**

Example (brief recap): *You are an idiot*

BoW: 'You', 'are', 'an', 'idiot'

Word 3-grams: 'You are an', 'are an idiot'

Character 3-grams (char): 'You', 'ou ', 'u a', ' ar', 'are', ...

Character 3-grams (char_wb): ' Yo', 'You', 'ou ', ' ar', 'are', 're ', ...

analyzer='char_wb' in sklearn: character 3-grams only from text inside word boundaries; n-grams at the edges of words are padded with space.

Feature combination

Using hstack:

```
from sklearn.feature_extraction.text import CountVectorizer
from scipy.sparse import hstack
from sklearn.svm import LinearSVC
from sklearn.metrics import classification_report

vec_word = CountVectorizer(analyzer='word', ngram_range=(1, 1), lowercase=False)
vec_char = CountVectorizer(analyzer='char', ngram_range=(3, 3), lowercase=False)

X_train = hstack((vec_word.fit_transform(train.text), vec_char.fit_transform(train.text)))
X_test = hstack((vec_word.transform(test.text), vec_char.transform(test.text)))

Y_train = train.labels
Y_test = test.labels

clf_svc = LinearSVC(random_state=0)
clf_svc.fit(X_train, Y_train)

predicted = clf_svc.predict(X_test)
print(classification_report(Y_test, predicted))
```

Feature combination

Using Pipeline and FeatureUnion:

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.pipeline import Pipeline, FeatureUnion
from sklearn.svm import LinearSVC
from sklearn.metrics import classification_report

vec_word = CountVectorizer(analyzer='word', ngram_range=(1, 1), lowercase=False)
vec_char = CountVectorizer(analyzer='char', ngram_range=(3, 3), lowercase=False)

word_char = FeatureUnion([('word', vec_word), ('char', vec_char)])
pipe = Pipeline([('vec', word_char), ('clf', LinearSVC(random_state=0))])

X_train = train.text
Y_train = train.labels

X_test = test.text
Y_test = test.labels

pipe.fit(X_train, Y_train)

predicted = pipe.predict(X_test)
print(classification_report(Y_test, predicted))
```

Commonly-used pre-processing steps

Pre-processing steps depend on the data source (social media platforms, blogs, others)

Some of the pre-processing steps often used for hate speech detection:

- Social media data
 - User mentions → replace by a placeholder/remove
Example: *@Isa and @Ilia* → *@User and @User*
 - URLs → placeholder/remove
 - Hashtags → placeholder/remove
 - Emojis → placeholder/remove
 - ...
- Other data sources
 - Function words → remove (!)
 - Punctuation → remove (!)
 - ...

Commonly-used machine learning algorithms and weighting schemes

ML algorithms often used for hate speech detection (and example papers):

- Support Vector Machines ([Caselli et al., 2021](#))
- Logistic Regression ([Kurrek et al., 2022](#))
- (Multinomial) Naive Bayes ([Pandey et al., 2022](#))
- Random Forest ([Nugroho et al., 2019](#))
- Decision Trees ([Putri et al., 2017](#))
- ...

Weighting schemes:

- tf (CountVectorizer)
- tf-idf (TfidfVectorizer)

Other features: LIWC features

Linguistic Inquiry and Word Count (LIWC): is a text analysis program that calculates the percentage of words in a given text that fall into various linguistic, psychological and topical categories indicating various social, cognitive, and affective processes.

The LIWC-22 dictionary ([Boyd et al., 2022](#)):

- Composed of over 12,000 words, word stems, phrases, and select emoticons.
- Each dictionary entry is part of one or more categories, or subdictionaries, designed to assess various psychosocial constructs.
- E.g., the word *cried* is part of 10-word categories: affect, tone_pos, emotion, emo_neg, emo_sad, verbs, focuspast, communication, linguistic, and cognition

Other features: sentiment- / emotion-based features

NRC emotion lexicon ([Mohammad and Turney, 2013](#)): 14,182 emotion words and their associations with eight emotions and two sentiments.

E.g., *illness*

Emotion/ sentiment	Binary association
anger	0
fear	1
anticipation	0
trust	0
surprise	0
sadness	1
joy	0
disgust	0
positive	0
negative	1

Gao and Huang (2017): LIWC and NRC features

Gao and Huang (2017):

- Character n-grams ($n = 2-4$)
- Word n-grams ($n = 1-2$)
- LIWC features:
 - Used 125 semantic categories from LIWC (version 2015)
 - Each word is converted into a 125 dimension LIWC vector, one dimension per semantic category
- NRC features:
 - Each word is converted into a 10 dimension emotion vector, corresponding to eight emotion types and two polarity labels

Classifier: logistics regression

Dataset: the Fox News User Comments corpus (discussion threads on Fox News articles)

Gao and Huang (2017): results

Results (binary classification: hateful or non-hateful):

Features	Input Contents	Accuracy	Precision	Recall	F1	AUC
char (baseline)	comment	0.738	0.549	0.469	0.504	0.733
+word	comment	0.735	0.548	0.443	0.488	0.736
+LIWC+NRC	comment	0.732	0.533	0.465	0.495	0.740
+word+LIWC+NRC	comment	0.747	0.568	0.476	0.517	0.750

Stylometric and emotion-based approach: features

Stylometric and emotion-based approach (Markov et al., 2021)

Features:

- Part-of-speech (POS) tags: morpho-syntactic patterns
- Function words (FWs): stylometric patterns
- Emotion-based features: emotion-conveying words, their frequency, and emotion associations from the NRC emotion lexicon (Mohammad and Turney, 2013)

Mental illness on parade

POS: ADJ NOUN ADP NOUN

POS & FWs: ADJ NOUN on NOUN

POS & FWs & emotions: ADJ illness on parade & 2 & fear sadness...

- Word and character n-grams
- + SVM + tf weighting scheme

Stylometric and emotion-based approach: results

Data: Facebook comments on migrants and LGBT topics

- FRENK dataset ([Ljubešić et al. 2019](#)) for English and Slovene
- LiLaH dataset for Dutch

Results for the binary classification task (hate speech vs. non-hate speech):

Model	English			Slovene			Dutch		
	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score
Random baseline	50.7	50.7	50.7	50.9	50.9	50.9	48.3	48.3	48.3
(1) BoW	71.0	70.8	70.9	68.5	68.5	68.5	72.0	70.9	71.1
(2) Char 1–3-grams	69.0	69.2	69.1	72.1	72.1	72.1	74.5	73.4	73.7
(3) BoW & char	70.6	70.6	70.6	72.4	72.4	72.4	75.0	74.4	74.6
(4) CNN	73.4	73.6	73.5	67.7	67.7	67.7	72.6	72.9	72.5
(5) LSTM	71.0	69.9	70.4	68.5	67.3	67.1	70.5	70.5	70.5
(6) BERT	74.9	74.6	74.8	73.0	72.9	72.9	74.3	74.1	74.2
(7) POS	57.3	57.0	57.1	63.2	63.1	62.8	63.9	62.9	62.9
(8) POS & FW	64.3	63.6	63.8	63.5	63.4	63.1	70.2	67.7	67.8
(9) POS & FW & emo	70.9	69.9	70.3	68.0	68.0	67.8	73.1	70.6	70.8
(10) POS & FW & emo & BoW & char	74.4	73.7	74.0	74.3	74.3	74.3	75.1	74.5	74.7

Stylometric and emotion-based approach

Markov and Daelemans (2021):

- Hate speech lexicon entries (POW lexicon) as additional feature vectors
- Character n-grams (n = 1–6, min_df = 10: appear in 10 training messages)
- SVM, tf-idf

Results for the binary classification task (hate speech vs. non-hate speech) on the FRENK (Ljubešić et al. 2019) and OLID (Zampieri et al., 2019) datasets:

Model	FRENK			OLID		
	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

Other features: syntactic features

Dependency relations for encoding syntactic information (Burnap and Williams, 2014):

"Totally fed up with the way this country has turned into a haven for terrorists. Send them all back home".

The typed dependency parser returns the following output:

```
[root(ROOT-0, Send-1), nsubj(home-5, them-2), det(home-5, all-3), amod(home-5, back-4),  
xcomp(Send-1, home-5)]
```

Rarely used: parsing social media produces errors

Other features: conversational (user) features

- **User demographics:** age, location, and gender (e.g., age extracted from user disclosures, gender from user names).
- **User history:** patterns in user behaviour, including daily logins, and posting history.
- **User profiles:** profile metadata as a proxy for digital identity, including usernames, user anonymity, the presence of updated profile pictures, biographies, verified account status, counts for followers and friends.
- **User networks:** e.g., interaction and connection-based social graphs.

Other features: community context

[Kurrek et al. \(2022\)](#): Enriching Abusive Language Detection with Community Context

- Capture information about community environment where online conversations take place
- Corpus: Slur-Corpus ([Kurrek et al., 2020](#)); 40k human-annotated Reddit comments
- Models: logistic regression, BERT
- Additional feature for the name of each subreddit that comments are sourced from.

Community context: results

	Performance			
	Accuracy	Precision	Recall	F1
PERSPECTIVE	0.6132	0.6147	0.6102	0.6079
LOG-REG	0.8003	0.8009	0.7994	0.7997
LOG-REG-COMM	0.8002	0.8001	0.7999	0.8000
BERT	0.8856	0.8854	0.8857	0.8855
BERT-COMM	0.8905	0.8904	0.8908	0.8905
BERT-COMM-SEP	0.8930	0.8930	0.8934	0.8930
BERT-COMM-NGH	0.8923	0.8924	0.8928	0.8923

Improvement due to the reduces false positive rates.

More features

- Emoticons
- Punctuation (e.g., question marks, exclamation marks)
- Length of the text: number of words / characters
- Proportion of uppercase characters to lowercase characters
- Negation words
- Action verbs (kill, destroy, ...)
- Personal pronouns (to distinguish between 'us' and 'them')
- Sentiment polarity of the text ([VADER sentiment analysis tool](#))
- ...

Conventional machine learning approaches: summary

- '+' Near state-of-the-art performance
- '+' Possibility for an explicit feature engineering
- '+' Preserve some sort of explainability
- '−' Need for feature engineering

Neural Networks

Conventional machine learning approaches: manually selected features are combined into input vectors and directly used for classification

(Deep) neural networks: automatically learn abstract features above input features

- **CNNs:** detect specific combinations of features; on character level, CNNs can deal with obfuscation of words
- **LSTMs:** take a sequence of words as input; powerful in capturing phrases and complex context information
- **BiLSTMs:** the input sequence is processed with correct and reverse order of words; long range dependencies

Examples for hate speech detection: [Meyer and Gambäck \(2019\)](#), [Chakrabarty et al. \(2019\)](#), [Modha et al. \(2018\)](#)

CNNs architecture: example

Layer (type)	Output Shape	Param #
embedding (Embedding)	(None, 300, 320)	6596160
dropout (Dropout)	(None, 300, 320)	0
conv1d (Conv1D)	(None, 298, 128)	123008
global_average_pooling1d (G1	(None, 128)	0
dropout_1 (Dropout)	(None, 128)	0
dense (Dense)	(None, 20)	2580
dropout_2 (Dropout)	(None, 20)	0
dense_1 (Dense)	(None, 1)	21
Total params: 6,721,769		
Trainable params: 6,721,769		
Non-trainable params: 0		

Source: [Markov and Daelemans \(2021\)](#)

Result: 79.4% F1-score (macro) on the OLID dataset.

LSTMs architecture: example

Layer (type)	Output Shape	Param #
embedding_12 (Embedding)	(None, 20, 300)	5644800
lstm_12 (LSTM)	(None, 300)	721200
dense_12 (Dense)	(None, 1)	301
Total params: 6,366,301		
Trainable params: 6,366,301		
Non-trainable params: 0		

Source: [Markov and Daelemans \(2021\)](#)

Result: 76.4% F1-score (macro) on the OLID dataset.

Neural networks: summary

- '+' Good performance
- '+' No need for feature engineering
- '-' Data hungry
- '-' Computationally expensive
- '-' Explainability

Transformer-Based Pre-Trained Language Models

Transformer-based pre-trained language models

Pre-training:

- Pre-trained on a large amount of unlabelled data (billions of tokens):
 - Masked language modelling (MLM) task: 15% of tokens are randomly masked, and then the model is trained to predict those tokens
 - Next sentence prediction (NSP) task: a binary classification task to predict if the second sentence succeeds the first sentence in the corpus

Fine-tuning:

- Fine-tuned on labelled data for downstream tasks, such as hate speech detection

Transformer-based pre-trained language models

General-purpose pre-trained language models:

- **BERT**: Bidirectional Encoder Representations from Transformers ([Devlin et al., 2019](#))
- **RoBERTa**: Robustly Optimized BERT Pretraining Approach ([Liu et al., 2019](#))
- **XLNet**: Generalized Autoregressive Pretraining for Language Understanding ([Yang et al., 2020](#))
- **ALBERT**: A Lite BERT for Self-supervised Learning of Language Representations ([Lan et al., 2020](#))

Transformer-based models: implementation

[Huggingface.co](#) has made using transformers-based models convenient with their Transformers API.

[Simple Transformers](#): NLP library designed to simplify the usage of transformer-based models. Built on Hugging Face and their Transformers library.

- Run your notebook on [colab](#)
- Install the [Simple Transformers](#) library: `!pip install simpletransformers`
- Follow the [documentation](#) to load a pre-trained language model
- Fine-tune the model on the training data and make predictions on the test data

Transformer-based models: example

```
!pip install simpletransformers
from simpletransformers.classification import ClassificationModel
from sklearn.metrics import classification_report

''' load the model'''
model = ClassificationModel('bert', 'bert-base-cased',
                           args={'reprocess_input_data': True,
                                'overwrite_output_dir': True}, use_cuda=True)

model.train_model(train) #fine-tune the model on the training data
predicted, probabilities = model.predict(test.text.to_list()) #predict test data
print(classification_report(test.labels, predicted)) #results

!cp -R outputs/checkpoint-1655-epoch-1 drive/MyDrive/ #save checkpoint to drive
```

Note: if you optimize the hyperparameters, set aside a subset of the training data as your evaluation (or development) set

Domain-specific transformer-based models: HateBERT

HateBERT: Retraining BERT for Abusive Language Detection in English (Caselli et al., 2020)

Retrained on 1.5 million Reddit comments from communities banned for being offensive, abusive, or hateful by applying the MLM objective.

Evaluated on 3 Twitter datasets for offensive (OLID/OffensEval-2019), abusive language (AbuseEval) and hate speech (HatEval)

Results:

Dataset	Model	Macro F1 Pos. class - F1	
OffensEval 2019	BERT	.803±.006	.715±.009
	HateBERT	.809±.008	.723±.012
	<i>Best</i>	.829	.752
AbusEval	BERT	.727±.008	.552±.012
	HateBERT	.765±.006	.623±.010
	Caselli et al. (2020)	.716±.034	.531
HatEval	BERT	.480±.008	.633±.002
	HateBERT	.516±.007	.645±.001
	<i>Best</i>	.651	.673

Huggingface: <https://huggingface.co/GroNLP/hateBERT>

Domain-specific transformer-based models: fBERT

fBERT: A Neural Transformer for Identifying Offensive Content (Sarkar et al., 2021)

Retrained using the MLM objective on 1.4 million offensive instances from the SOLID dataset (Rosenthal et al., 2021)

Evaluated on 3 Twitter datasets: HatEval, OLID, HS & O: Hate Speech and Offensive Language Detection.

Results:

Dataset	Model	Macro F1
HatEval	fBERT	0.596
	HateBERT	0.525
	BERT	0.483
OLID	fBERT	0.813
	HateBERT	0.801
	BERT	0.794
HS & O	fBERT	0.878
	HateBERT	0.846
	BERT	0.806

Huggingface: <https://huggingface.co/diptanu/fBERT>

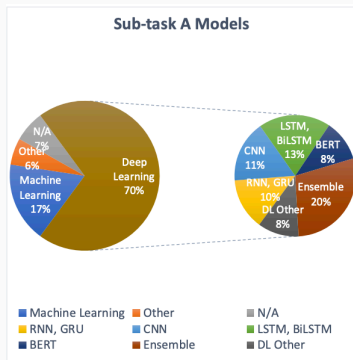
Transformer-based models: summary

- '+' State-of-the-art performance (Facebook uses RoBERTa)
- '+' No need for feature engineering
- '-' Data hungry
- '-' Computationally expensive
- '-' Explainability

OffensEval-2019 shared task: approaches

SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval) ([Zampieri et al., 2019](#))

Deep learning was the most popular approach, as were also ensemble models.



OffensEval-2019 shared task: results

Among the top-10 teams, seven used BERT (often as part of ensembles)

Sub-task A	
Team Ranks	F1 Range
1	0.829
2	0.815
3	0.814
4	0.808
5	0.807
6	0.806
7	0.804
8	0.803
9	0.802
CNN	0.800
10	0.798
11-12	.793-.794
13-23	.782-.789
24-27	.772-.779
28-31	.765-.768
32-40	.750-.759
BiLSTM	0.750
41-45	.740-.749
46-57	.730-.739
58-63	.721-.729
64-71	.713-.719
72-74	.704-.709
SVM	0.690
75-89	.619-.699
90-96	.500-.590
97-103	.422-.492
All NOT	0.420
All OFF	0.220
104	0.171

Summary

- **Lexicon-based approaches** rely on using external resources, e.g., offensive word lists: $\approx 70\%$ – 75% F1-score*
- **Conventional machine learning approaches:** SVM with word, character n-grams, other: $\approx 70\%$ – 78%
- **Deep neural networks:** CNNs, LSTMs, BiLSTMs: $\approx 75\%$ – 79% ;
- **Transformer-based approaches:** BERT, RoBERTa $> 80\%$; $\approx 82\%$
- **Ensemble learning:** $\approx 85\%$

Each approach has advantages and disadvantages

- **Performance:**
 - depends on the language, training data, how the task is framed
 - cross-domain drop is about 10%
 - high results considering the inter-annotator agreement

*Results for the binary hate speech detection task (hate speech VS. non-hate speech)

Questions?

Assignment 3