EMBEDDIA

Cross-Lingual Embeddings for Less-Represented Languages in European News Media

Research and Innovation Action
Call: H2020-ICT-2018-1
Call topic: ICT-29-2018 A multilingual Next generation Internet
Project start: 1 January 2019 Project duration: 36 months

D1.8: Final deep network architecture (T1.3)

Executive summary

This deliverable presents the advances of state-of-the-art neural network technologies, which were achieved within T1.3 of the EMBEDDIA project. First, we enhance LSTM and transformer networks, such as BERT, with morphological information in the form of universal POS tags and universal features. The approach is tested on nine EMBEDDIA languages using three tasks: named entity recognition, dependency parsing, and in connection with WP3, also comment filtering. As the information on the reliability of predictions is crucial in tasks that affect human users, we describe a Bayesian method to obtain well-calibrated reliability estimates from transformer networks and test it on the hate speech detection problem in several languages. We also demonstrate that a Bayesian ensemble of reliability estimates can improve text annotation. Finally, addressing a strategic need for merging different information sources in machine learning, we develop a unifying framework for data transformations into numerical vector space using deep neural networks. We show that propositionalization, used in relational learning, and embeddings, used in text and graph learning, can be viewed as two sides of the same coin.

Partner in charge: UL

Project co-funded by the European Commission within Horizon 2020 Dissemination Level		
PU	Public	PU
PP	Restricted to other programme participants (including the Commission Services)	–
RE	Restricted to a group specified by the Consortium (including the Commission Services)	–
CO	Confidential, only for members of the Consortium (including the Commission Services)	–

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825153

Deliverable Information

Document administrative information	
Project acronym:	**EMBEDDIA**
Project number:	**825153**
Deliverable number:	**D1.8**
Deliverable full title:	**Final deep network architecture**
Deliverable short title:	**Final deep networks**
Document identifier:	**EMBEDDIA-D18-FinalDeepNetworks-T13-submitted**
Lead partner short name:	**UL**
Report version:	**submitted**
Report submission date:	**31/12/2020**
Dissemination level:	**PU**
Nature:	**R = Report**
Lead author(s):	**Marko Robnik-Šikonja (UL), Matej Klemen (UL)**
Co-author(s):	**Kristian Miok (UL), Luka Krsnik (UL), Blaž Škrlj (JSI), Nada Lavrač (JSI)**
Status:	**__ draft, __ final, _x_ submitted**

The EMBEDDIA Consortium partner responsible for this deliverable has addressed all comments received. Changes to this document are detailed in the change log table below.

Change log

Date	Version number	Author/Editor	Summary of changes made
19/01/2020	v1.0	Marko Robnik-Šikonja (UL)	First outline.
20/11/2020	v1.1	Marko Robnik-Šikonja (UL), Luka Krsnik (UL), Matej Klemen (UL)	Compilation of results.
23/11/2020	v1.2	Marko Robnik-Šikonja (UL), Kristian Miok (UL), Blaž Škrlj (JSI), Nada Lavrač (JSI)	Compilation of results.
27/11/2020	v1.3	Marko Robnik-Šikonja (UL)	Integration of results
02/12/2020	v1.4	Matej Martinc (JSI)	Internal review.
06/12/2020	v1.5	Marko Robnik-Šikonja (UL)	Taking the review into account.
10/12/2020	v1.6	Saturnino Luz (UEDIN)	Internal review.
13/12/2020	v1.7	Marko Robnik-Šikonja (UL)	Taking the review into account.
15/12/2020	v1.8	Nada Lavrač (JSI)	Report quality checked and finalised.
19/12/2020	final	Marko Robnik-Šikonja (UL)	Final corrections.
29/12/2020	submitted	Tina Anžič (JSI)	Report submitted.

Table of Contents

List of abbreviations

BAN	Bayesian Attention Networks
BERT	Bidirectional Encoder Representations from Transformers
BNN	Bayesian Neural Networks
CF	Comment Filtering
DNN	Deep Neural Networks
DP	Dependency Parsing
ELMo	Embeddings from Language Models
ILP	Inductive Logic Programming
LSTM	Long Short-Term Memory network
MCD	Monte Carlo Dropout
MCD BERT	Monte Carlo Dropout for BERT
MM	Multivariate Normal Mixture Conditional Likelihood Model
NE	Named Entity
NER	Named Entity Recognition
NLP	Natural Language Processing
POS	Part-Of-Speech
RL	Relational Learning
RNN	Recurrent Neural Network
SVM	Support Vector machines

1 Introduction

The main objective of task T1.3 of the EMBEDDIA project is to advance deep learning technology, emphasizing morphologically rich, less-resourced languages. This report describes the results of the work performed in T1.3 from M13 till M24. The initial work within T1.3 from M1 to M12 was reported and accepted as deliverable D1.4 in M12. The work presented in this deliverable uses cross-lingual mappings developed in T1.1 and contextual embeddings developed in T1.2.

While deep neural networks (DNNs) combined with embeddings have revolutionised modern NLP approaches, the choice of architecture is still an important consideration. Many options are available, e.g., recurrent neural networks (RNN), long-short-term memory networks (LSTM), convolutional neural networks (CNN), and transformer networks, but they are mostly tested for English, while less-resourced languages are under-researched. This is especially true for morphologically rich languages, including all Slavic and Baltic languages, covered in the EMBEDDIA project. Many of these languages use a single inflectional morpheme to denote multiple grammatical, syntactic, or semantic features. Adding such explicit morphological information to a deep neural network can be highly beneficial [Soricut and Och, 2015, Kim et al., 2016]. In T1.3, we tested adding morphological information to two state-of-the-art neural network architectures: LSTM and transformers-based BERT models. We combined these networks with universal POS tags and universal morphological features. We tested the modified architectures on three tasks, available across several languages: named entity recognition (NER), dependency parsing (DP), and comment filtering (CF). The contributions to the NER and DP tasks are relevant for WP2, which develops text enrichment technologies. At the same time, the improvements in CF contribute to WP3, where user-generated contents are analysed.

The information on the reliability of predictions made by machine learning models is crucial in tasks that affect humans users, such as the ones addressed in WP3 (e.g., comment filtering and hate speech prediction), WP4 (e.g., text annotation), and T1.4 (explanation and visualisation of predictions). To address these needs, we present our work on obtaining reliability scores for DNN predictions. We propose a Bayesian method using Monte Carlo Dropout (MCD) within the attention layers of the transformer models, such as BERT, to provide well-calibrated reliability estimates. We evaluate and visualise the results of the proposed approach in hate speech detection problems in several languages. Additionally, we describe how this approach can be used to simplify the text annotation process. This work depends on the embeddings and cross-lingual maps developed in T1.1 and T1.2 and is relevant for T1.4, WP3, and WP4.

Finally, there is a strategic need for merging different information sources in machine learning. This would overcome the weakness that even state-of-the-art DNNs can only solve tasks in individual, narrowly defined domains. Due to the success of DNNs, it makes sense to transform data of many different modalities (such as text, relational data, electronic health records, etc.) to a tabular form, suitable for processing with DNNs. For that purpose, we present a unifying framework for data transformations into a numerical vector space. We show that propositionalization, used in relational learning, and embeddings, used in text and graph learning, can be viewed as two sides of the same coin. We propose two efficient implementations of the unifying methodology: an instance-based PropDRM approach, and a feature-based PropStar approach based on StarSpace embeddings. The results demonstrate that the new algorithms can outperform existing relational learners and can solve much larger problems in terms of dataset size.

The main contributions presented in this report (in the order of appearance) are as follows.

1. Extensions of LSTM neural networks and the state-of-the-art BERT models with explicit morphology (POS tags, morphological features), described in Section 2, and the paper by Klemen et al. [2020] (submitted to the Natural Language Engineering journal), included in Appendix A.
2. Extensions of state-of-the-art transformer neural architectures for text processing with reliability scores, described in Section 3. The final version of this work was accepted for publication in the Cognitive Computation journal [Miok et al., 2021] and is included in the Appendix B, while the initial

version was published in the ICML UDL workshop [Miok et al., 2020b] and is included in Appendix C.

3. Two semi-supervised prediction reliability-based methods to guide the annotation process: a Bayesian deep learning model and a Bayesian ensemble method. These methods are described in Section 4 and the paper by Miok et al. [2020a], published in the COLING 2020 Linguistic Annotation Workshop and included in Appendix D.
4. A novel unifying framework for data transformations into numerical vector space with two efficient implementations of the unifying methodology, described in Section 5, and the paper by Lavrač et al. [2020], published in the Machine Learning journal and included in Appendix E.

We present conclusions about the tested extensions to deep neural networks and other contributions of T1.3 in Section 6 where we also outline possibilities for further work. Availability of the new resources produced in this work is presented in Section 7.

2 Adaptations of neural networks for morphologically rich languages

Deep learning for processing natural language is becoming a standard, with excellent results in a diverse range of tasks. Two state-of-the-art architectures for text-related modelling are long short-term memory (LSTM) networks [Hochreiter and Schmidhuber, 1997] and transformers [Vaswani et al., 2017]. LSTMs are recurrent neural networks that process the text sequentially, meaning that they process text one token at a time, building up its internal representation in the network's hidden layers. Due to the recurrent nature of LSTM, which degrades the efficiency of parallel processing, as well as demonstrated improvements in performance using transformer models, models based on the transformer architecture are gradually replacing LSTMs across many tasks. Transformers can process text in parallel, using self-attention and positional embeddings to model the text's sequential nature.

A common trend in using transformers is to first pre-train them on large monolingual corpora with abstract, general-purpose objective, and then fine-tune them for a specific task, such as text classification. For example, the BERT (Bidirectional Encoder Representations from Transformers) architecture [Devlin et al., 2019] uses transformers and is pretrained with masked language modelling and order of sentences prediction tasks to build a general language understanding model. During the fine-tuning for a specific downstream task, additional layers are added to the BERT model. The model is trained on specific data to capture the specific knowledge required to perform the task.

Most of the research in the natural language processing (NLP) area focuses on English, ignoring the fact that English is specific in terms of the low amount of information expressed through morphology (English is a so-called analytical language). In our work, we focus on adapting modern deep neural networks, namely LSTMs and BERT, for several morphologically rich languages by explicitly including the morphological information. The languages we analyse contain rich information about grammatical relations in the morphology of words instead of in particles or relative positions of words (as is the case in English). For comparison, we also evaluate our models in English. Although previous research has shown that the state of the art methods such as BERT already capture some information contained in the morphology [Pires et al., 2019, Edmiston, 2020], our experiments involve several languages with rich morphology, where neural networks could benefit from explicit morphological features.

Specifically, we present methods that combine BERT with separately encoded morphological properties: universal part of speech tags (uPOS tags) and universal features (grammatical gender, tense, conjugation, declination, etc). We evaluate them on three downstream tasks: named-entity recognition (NER), dependency parsing (DP), and comment filtering (CF). We perform similar experiments on LSTM networks and compare the results for both architectures. Besides English, for each task, we analyse the largest possible subsets of eight EMBEDDIA languages: Croatian, Estonian, Finnish, Latvian, Lithua-

nian, Russian, Slovene, and Swedish. The choice of a subset of these languages reflects the availability of sufficient resources (datasets, embeddings, and corpora).

Our experiments show that the addition of morphological features has mixed effects depending on the task. Across the tasks where the added morphological features improve the performance, we show that (a) they benefit the LSTM-based models even if the features are noisy, and (b) they benefit the BERT-based models only when the features are of high quality (i.e. human checked), suggesting that BERT models already capture the morphology of the language; however, there is room for improvement either in designing pre-training objectives that can capture these properties or when high-quality features are available.

In the subsections below, we first describe the used datasets and their properties, followed by an outline of models with additional morphological information and their performance.

2.1 Data

This section describes the datasets used in our experiments, separately for each of the three tasks: NER, DP, and CF.

In the NER experiments, we use datasets in eight languages: Croatian, English, Estonian, Finnish, Latvian, Russian, Slovene, and Swedish. We omit the Lithuanian language from this experiment, as models for obtaining POS tags and universal features for this language were not available in the used tools at the time of performing the experiments. The label sets used in datasets for different languages vary, meaning that some contain more fine-grained labels than others. To make results across different languages consistent, we trim labels in all datasets to the four common ones: location (LOC), organization (ORG), person (PER), and "no entity" (OTHR).

To test morphological neural networks on the DP task, we used datasets in nine languages (Croatian, English, Estonian, Finnish, Latvian, Lithuanian, Russian, Slovene, and Swedish). The datasets are obtained from the Universal Dependencies [Nivre et al., 2020].

While there exist comparable datasets across different languages for the NER and DP task, no such standard datasets are available for the CF task. For that reason, in our experiments on CF, we used two languages with adequate datasets: English and Croatian. For English experiments, we used a subset of toxic comments from Wikipedia's talk page edits[1]. The comments are annotated with six possible labels: toxic, severe toxic, obscene language, threats, insults, and identity hate (making a total of six binary target variables). As we concentrate on hate speech, we extracted comments from four categories: toxic, severe toxic, threats, and identity hate; a total of $21,541$ instances. We randomly chose the same amount of comments that do not fall in any of the mentioned categories, obtaining the final dataset of $43,082$ instances. For Croatian language experiments, we formed a dataset from user comments published in the Croatian news site 24sata. The comments used are either non-problematic or labelled with one of the eight rules they break. We extracted all the $17,868$ comments from the third category (the hate speech label) and approximately the same number of comments that do not break any rules. Our final dataset contains $35,635$ instances.

2.2 Neural networks with morphological features

In this section, we describe the architectures of neural networks used in our experiments. Their common property is that we enhance standard word embeddings based inputs with embeddings of morphological features. We work with recent successful neural network architectures, LSTM and transformers (i.e. BERT) models. A detailed description of architectures is available in the following subsections, separately for each of the three evaluation tasks. For each task and architecture, we describe the baseline architecture and the enhanced one.

[1] https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/data

2.2.1 Named entity recognition models

In the NER task, we use two baseline neural networks (LSTM and BERT) and the same two models with additional morphological information: POS tag embeddings and universal feature embeddings. The baseline models and their enhancements are displayed in Figure 1.

The first baseline model (left-hand side of Figure 1) is a unidirectional (left-to-right) LSTM model, which takes as an input a sequence of tokens, embedded using 300-dimensional fastText embeddings [Bojanowski et al., 2017], which are particularly suitable for morphologically rich languages as they work with subword inputs[2]. For each input token, its LSTM hidden state is extracted and passed through the linear layer to compute its NER score (probability for each of the four NER labels).

The second baseline model (right-hand side of Figure 1) is the multilingual BERT model (the base cased variant). In our experiments, we follow the sequence tagging approach suggested by the authors of BERT [Devlin et al., 2019]. Input sequences, starting with the special [CLS] token and ending with the [SEP] token, are passed through the BERT model. The output of the last BERT hidden layer is passed through the linear layer to obtain the predictions for NER labels.

Both baseline models (LSTM and BERT) are enhanced with the same morphological information: POS tag embeddings and universal feature embeddings for each input token. We embed the POS tags using 15-dimensional embeddings. For each of the 23 universal features used (we omitted the *Typo* feature, as the version of the POS tagger we used did not annotate this feature), we constructed 15-dimensional embeddings. We computed the POS tags and morphological features using the Stanza [Qi et al., 2020] system in the universal dependencies mode. In the enhanced architectures, we included another linear layer before the final linear classification layer to model possible interactions.

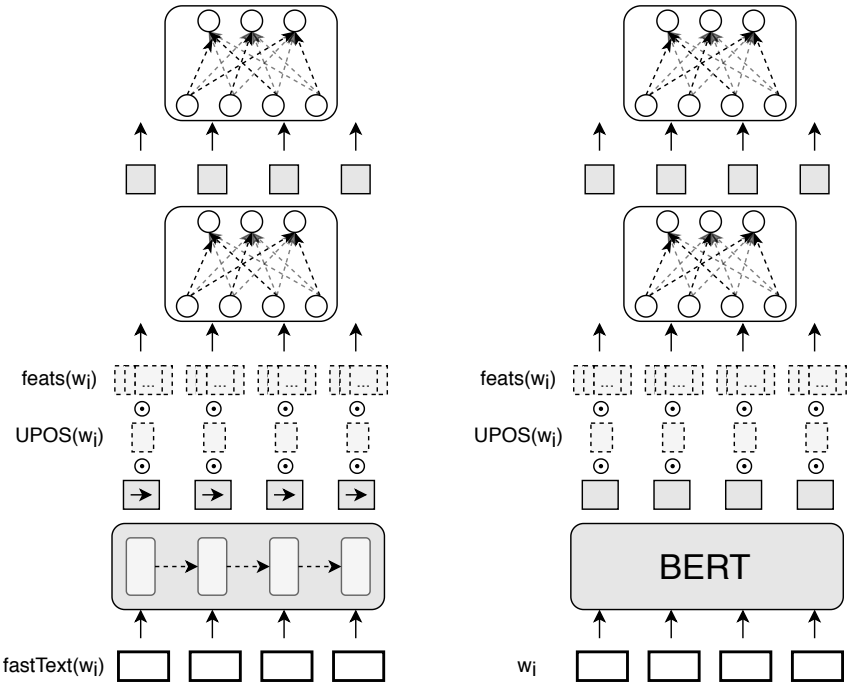


Figure 1: The baseline LSTM-based (left) and BERT-based (right) models for the NER task along with our modifications with morphological information. The dotted border of UPOS (Universal Part-of-Speech) vectors, morphological features (feats), and the linear layer marks that their use is optional and varies across experiments. The $\odot$ symbol between layers represents the concatenation operation. The w_i symbol stands for token i; in case of LSTM, tokens enter the model sequentially and we show the unrolled network, while BERT processes all tokens simultaneously.

[2]The precomputed embeddings are available at https://fasttext.cc/docs/en/crawl-vectors.html.

2.2.2 Dependency parsing models

As the baseline model in the DP task, we use the deep biaffine graph-based dependency parser [Dozat and Manning, 2016]. The enhancements with the morphological information are at the input level. The baseline model and its enhancements are shown in Figure 2.

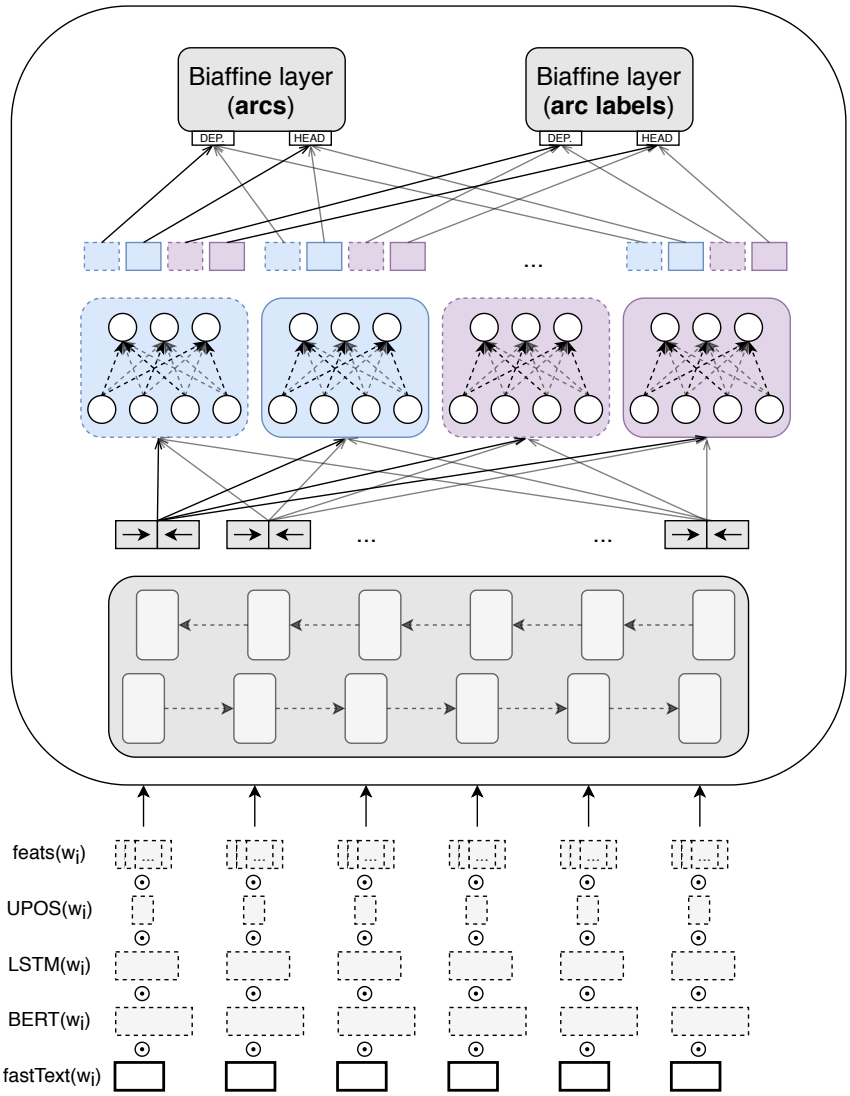


Figure 2: The deep biaffine graph-based dependency parser along with our enhancements at the input level. The dotted border of input embedding vectors, UPOS vectors, and morphological features (feats) is optional and varies across experiments. The $\odot$ symbol between layers represents the concatenation operation. The w_i symbol stands for token i; tokens enter the LSTM model sequentially, and we show the unrolled network.

The baseline parser combines a multi-layer bidirectional LSTM network with a biaffine attention mechanism to jointly optimize the prediction of arcs and arc labels. We leave most baseline architectural hyperparameters at values described in the original paper (3-layer bidirectional LSTM with 100-dimensional input word embeddings, and the hidden state size of 400).

In our experiments, we concatenate the non-contextual word embeddings with various types of additional information. The first additional input is contextual word embeddings, which we obtain either by using the hidden states of an additional single-layer unidirectional LSTM or by using a learned linear

combination of all hidden states of BERT. Although the LSTM layers are already present in the baseline parser, we included an additional layer at the input level to keep the experimental settings similar across our three evaluation tasks. The second additional input is universal POS embeddings (UPOS), and the third one are universal feature embeddings (feats). These embeddings are concatenated separately for each token in the sentences. The sizes of the additional LSTM layer, POS tag embeddings, and universal feature embeddings are treated as tunable hyperparameters. As the baseline input embeddings, we used pre-trained 100-dimensional fastText embeddings, which we obtained by reducing the dimensionality of the publicly available 300-dimensional vectors with the fastText's built-in dimensionality reduction tool.

In DP experiments, we used POS tags and morphological features from two sources. The first source is human annotations provided in the used datasets. The second source of morphological information is predictions of Stanza models [Qi et al., 2020]. These two types of annotations are used to assess the quality of morphological information; namely, we check if manual human annotations provide any benefit compared to automatically determined POS tags and features.

2.2.3 Comment filtering models

In the NER evaluation task, we add additional morphological information to standard LSTM and BERT models. The baseline models and enhanced models for this task are similar to those in the NER evaluation. However, we operate at the sequence level here as opposed to the token level in the NER task. The architecture of models is shown in Figure 3.

As baselines, we take a single layer unidirectional LSTM network (the top part of Figure 3) and the multilingual base uncased BERT model (the bottom part of Figure 3). The difference in the used BERT dialect (*uncased* as opposed to the *cased* in the NER task) is due to better performance detected in preliminary experiments on a separate Croatian validation set.

In the LSTM baseline model, the words of the input sequence are embedded using pre-trained 300-dimensional fastText embeddings. As the representation of the whole sequence, we take the output of the last hidden state, which then passes through the linear layer to obtain the prediction scores. In the BERT baseline model, we take the sequence classification approach suggested by the authors of BERT. The input sequence is prepended with the special [CLS] token and passed through BERT. The sequence representation corresponds to the output of the last BERT hidden layer for the [CLS] token, which is passed through a linear layer to obtain the prediction scores.

As in other tasks, we augment the baseline models with POS tags and universal feature embeddings. We obtain the embeddings for each token separately from UDPipe models [Straka and Straková, 2017]. The reason for not using Stanza in this set of experiments is that we started experimenting before its release. Additional testing with Stanza did not show any difference in performance, so we kept the UDPipe models for this set of experiments. We aggregate the obtained embeddings using three different pooling mechanisms: mean, weighted combination, or LSTM pooling. Given the POS tag or universal feature embeddings, the mean pooling outputs the mean of all token embeddings; the weighted pooling outputs the weighted combination of token embeddings, and the LSTM pooling outputs the last hidden state obtained by passing the sequence of embeddings through the LSTM network. Both the embedding sizes and the type of pooling are treated as tunable hyperparameters. The coefficients of the weighted combination are learned by projecting the sequence embeddings into a sequence of independent dimension values, which are normalized with the softmax. This compresses the embedding sequence, establishes its fixed length, and allows for different morphological properties to have a different impact on the sequence representation. This approach tests the contextual encoding of morphological properties. For example, adjectives might be assigned a higher weight than other POS tags due to more emotional content that often indicates insults.

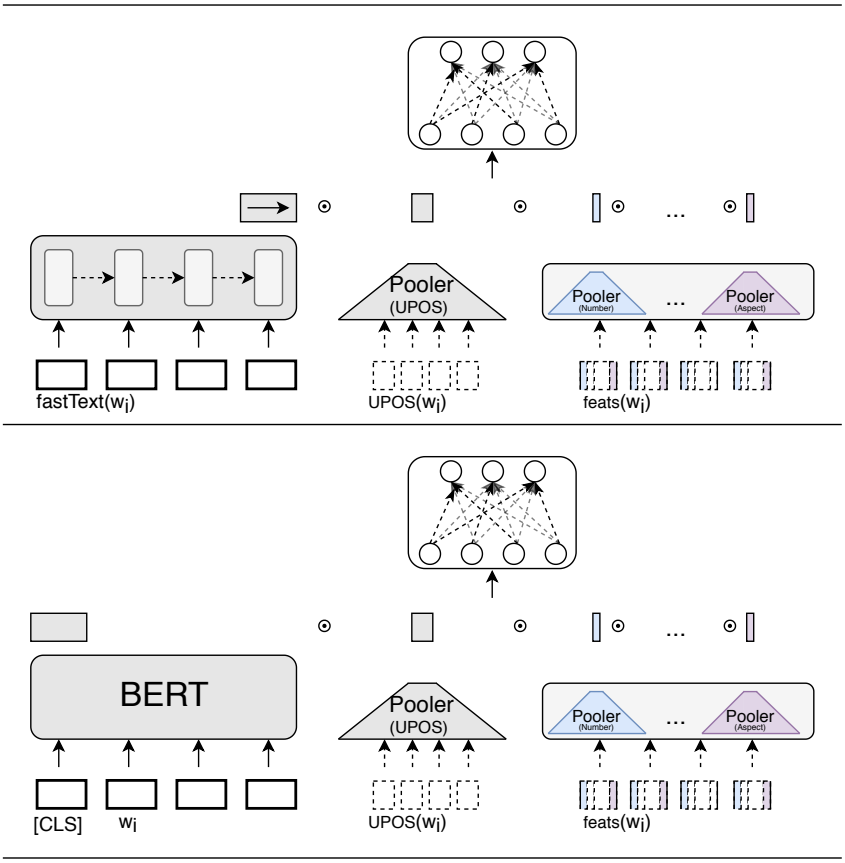


Figure 3: The baseline LSTM (top) and BERT (bottom) models for the NER task along with our modifications with morphological information. The dotted border of UPOS vectors and morphological features (feats) marks that their use is optional and varies across experiments. The $\odot$ symbol between layers represents the concatenation operation. The w_i symbol stands for token i; in case of LSTM, tokens enter the model sequentially and we show the unrolled network, while BERT processes all token simultaneously.

2.3 Experimental settings

This section presents the evaluation scenario for the three evaluation tasks, presented separately for each of the tasks. We start with the NER task, followed by the DP and NER tasks. The results are presented in Section 2.4.

2.3.1 Experimental settings for NER

For NER, we train each BERT model for 10 epochs and each LSTM model for 50 epochs. These parameters were determined during preliminary testing on the Slovene dataset. The selected numbers of epochs are chosen to balance the performance and training times of the models. All NER models are evaluated using 10-fold cross-validation.

We evaluate the models with the F_1 score, a harmonic mean of precision and recall measures. This measure is typically used in NER in a way that precision and recall are calculated separately for each of the three entity classes (location, organization, and person, but not ”no entity”). For each metric, we compute the weighted average over the class scores using the frequencies of class values in the datasets as the weights. Ignoring the ”no entity” (OTHR) label is a standard approach in the NER evaluation and disregards words that are not annotated with any of the named entity tags, i.e. the evaluation focuses on the named entities.

2.3.2 Experimental settings for dependency parsing

We train each DP model for a maximum of 10 epochs, using an early stopping tolerance of 5 epochs in the process. All models are evaluated using predefined splits into training, validation, and testing set, determined by the respective treebank authors and maintainers. The final models, evaluated on the test set, are selected based on the maximum mean of unlabelled (UAS) and labelled (LAS) attachment scores on the validation set. The UAS and LAS are standard accuracy metrics in DP. The UAS score is defined as the proportion of tokens that are assigned the correct syntactic head. In contrast, the LAS score is the proportion of tokens that are assigned the correct syntactic head as well as the dependency label [Jurafsky and Martin, 2009].

2.3.3 Experimental settings for comment filtering

In our NER experiments, we train BERT models for a maximum of 10 epochs and LSTM models for a maximum of 50 epochs, using early stopping with the tolerance of 5 epochs.

We evaluate the models using the classification accuracy metric. The choice reflects the fact that the distributions of class labels are approximately balanced in our datasets. We use fixed training, validation, and test sets, obtained by randomly dividing the datasets in the ratio 60%:20%:20%. The initial results in the Croatian hate speech experiments showed a large variance; therefore, we repeated each experiment five times. For this task and Croatian language, we report the obtained mean classification accuracy and its standard deviation.

2.4 Experimental results

In this section, we compare the results of baseline models with their enhancements using additional morphological information. According to the evaluation task, we split the presentation into three parts: NER, DP, and CF. To further analyse different aspects of the proposed morphological enhancements, we conducted two ablation studies on the DP task where the datasets and evaluation settings allow many experiments. We test the impact of morphological information quality and different variants of BERT models.

2.4.1 Results for the NER task

For the NER evaluation task, we present results of the baseline NER models and models enhanced with the POS tags and morphological features, as introduced in Section 2.2.1. Table 1 shows the results for LSTM and BERT models. We compute the statistical significance of the differences between the baseline LSTM models and their best-performing counterparts with morphological additions. We use Wilcoxon signed-rank test [Wilcoxon et al., 1970] and underline the statistically significant differences at $p = 0.01$ level. We do not perform the significance of differences tests for BERT as the differences for those models are marginal.

As expected, the baseline models involving BERT outperform their LSTM counterparts across all languages by a large margin. When adding POS tags or universal features to LSTM-based models, we observe a noticeable increase in performance over seven languages' baselines. For three of them (Croatian, Russian, and Swedish), the increase in F_1 score is under one percent. For three of them (Estonian, Finnish, and Slovene), the differences in performance are statistically significant, and the increase ranges from 1.4% (Finnish) to 4.6% (Slovene). The only language for which there is no improvement with the LSTM-based models is English.

In BERT-based models, the additional morphological features do not seem to make a practical difference. For all languages except Slovene, the increase in F_1 values over the baseline is under 0.5%.

However, since preliminary tests in this task were performed on Slovene, the larger observed increase might be due to manual overfitting.

Table 1: F_1 scores for different models on the NER task in different languages. The upper part of the table shows LSTM models and the lower part shows BERT models. The best scores for each language are marked with the bold typeface separately for LSTM and BERT models. Best scores for which the difference to the baseline is statistically significant are underlined.

Model	CRO	ENG	EST	FIN	LAT	RUS	SLO	SWE
fT + LSTM	0.700	**0.890**	0.769	0.814	0.573	0.752	0.651	0.785
fT + LSTM + UPOS	**0.708**	0.889	**0.792**	0.824	**0.590**	0.754	0.692	0.787
fT + LSTM + UPOS + feats	0.706	0.883	0.781	**0.828**	0.588	0.759	**0.697**	0.785
fT + LSTM + feats	0.697	0.885	0.777	0.826	0.581	**0.760**	0.669	**0.793**
BERT	0.874	0.948	0.875	0.926	0.766	0.868	0.848	0.885
BERT + UPOS	**0.875**	0.948	0.874	0.927	**0.773**	**0.870**	**0.857**	**0.886**
BERT + UPOS + feats	**0.875**	**0.949**	**0.879**	0.927	0.763	0.869	0.853	0.883
BERT + feats	0.874	0.947	0.874	**0.928**	0.769	0.869	0.851	0.878

2.4.2 Results for the dependency parsing task

For the DP evaluation task on different languages (described in Section 2.2.2), we present UAS and LAS accuracy in Tables 2 and 3, denoting maximum mean of unlabelled (UAS) and labelled (LAS) attachment scores on the validation set, respectively. We first show the scores of the baseline models (multi-layer bidirectional LSTM network with biaffine attention), followed by these models enhanced with additional inputs: LSTM or BERT contextual embeddings, POS tags, and morphological features, introduced in Section 2.2.2. The results in the upper part of Section 2.2.2 represent enhancements incorporating LSTM embeddings, and in the lower part, we show the enhancements including BERT embeddings. We also statistically test the differences in UAS and LAS scores between the best performing enhanced variants and their baselines without morphological additions. As the splits are fixed in the DP tasks, we use Z-test for the equality of two proportions [Kanji, 2006] at $p = 0.01$ level. The null hypothesis is that the scores of compared models are equal. For languages where the null hypothesis can be rejected, we underline the respective best result.

Like the other two evaluation tasks, the models involving BERT embeddings outperform the baselines involving LSTM embeddings on all languages by a large margin. Enhancements with LSTM embeddings improve over parsers without contextual embeddings. Models with added POS tags or universal features noticeably improve over baselines with only LSTM embeddings in both UAS and LAS scores for all languages. The increase ranges between 1.2% (Russian) and 6.8% (Lithuanian) for UAS and between 2.4% (Russian) and 10.42% (Lithuanian) for LAS. All compared differences between the LSTM baselines and the best enhanced variants are statistically significant at $p = 0.01$.

Contrary to the results observed on the other two tasks, the addition of morphological features to the baseline models with BERT embeddings improves the performance scores for all languages. For some languages, the increase seems to be practically insignificant (under one percent). However, for six languages out of nine, the increase in UAS is over one percent and statistically significant; the same is true for eight languages out of nine for the LAS score. For these languages, the improvement ranges from 1.1% (Finnish) to 1.8% (Lithuanian) for UAS, and from 1.1% (Croatian) to 4.4% (Lithuanian) for LAS. The only language for which both UAS and LAS scores are practically equal to BERT embeddings' baselines is Russian.

2.4.3 Results for the comment filtering task

Table 4 shows the NER results for LSTM and BERT baselines and their enhancements, described in Section 2.2.3. All BERT-based models outperform the LSTM-based models by a large margin for both

Table 2: UAS scores for different models on the DP task in different languages. The upper part of the table shows baseline parser with enhancements using LSTM-based contextual embeddings; the lower part shows baseline parser enhanced with variants of BERT embeddings. The best scores for each language are marked with the bold typeface separately for LSTM and BERT embeddings extensions. Statistically significant differences in best scores at $p = 0.01$ level are underlined.

Model	CRO	ENG	EST	FIN	LAT	LIT	RUS	SLO	SWE
baseline	84.83	83.54	79.35	82.45	81.51	70.51	83.04	86.66	80.68
+LSTM	86.48	84.87	81.92	84.13	83.63	71.82	85.02	86.92	84.03
+LSTM+UPOS	87.81	**87.26**	85.30	86.82	86.73	75.83	85.75	91.99	86.78
+LSTM+UPOS+f.	**87.82**	87.21	**86.12**	**87.64**	**87.97**	**78.62**	86.19	**92.42**	**87.19**
+LSTM+feats	87.52	85.97	84.31	85.23	86.09	78.10	**86.26**	90.67	84.90
+mBERT	91.72	91.43	88.02	89.99	87.87	79.79	90.32	93.66	90.16
+mBERT+UPOS	**92.17**	**91.64**	89.26	90.53	89.11	80.48	90.53	94.70	91.21
+mBERT+UPOS+f.	91.97	91.56	**89.51**	**91.09**	**89.63**	**81.64**	**90.65**	**94.95**	**91.46**
+mBERT+feats	91.72	91.35	88.32	90.33	88.92	81.61	90.64	94.42	90.64

Table 3: LAS scores for different models on the DP task in different languages. The upper part of the table shows baseline parser with enhancements using LSTM-based contextual embeddings; the lower part shows baseline parser with enhancements using BERT embeddings. The best scores for each language are marked with the bold typeface separately for LSTM and BERT embedding extensions. Statistically significant differences in best scores at $p = 0.01$ level are underlined.

Model	CRO	ENG	EST	FIN	LAT	LIT	RUS	SLO	SWE
baseline	77.55	79.28	73.17	77.09	75.77	62.23	77.22	82.12	75.06
+LSTM	79.59	80.69	76.24	78.60	78.08	63.65	79.24	83.03	78.97
+LSTM+UPOS	81.95	**84.68**	81.36	83.06	82.75	70.45	80.88	88.75	82.65
+LSTM+UPOS+f.	**82.84**	84.48	**83.46**	**83.97**	**84.27**	**74.07**	**81.68**	**90.32**	**83.17**
+LSTM+feats	82.00	82.96	81.00	81.16	81.81	72.40	81.64	88.38	80.31
+mBERT	86.37	88.09	84.30	86.07	83.02	72.62	85.99	91.67	86.67
+mBERT+UPOS	**87.63**	**89.35**	86.72	87.32	85.43	74.99	86.24	92.90	87.80
+mBERT+UPOS+f.	87.44	89.13	**87.19**	**87.99**	**86.16**	**77.04**	**86.56**	**93.40**	**88.38**
+mBERT+feats	86.81	88.81	85.72	87.07	85.19	76.33	86.37	92.74	87.25

languages, Croatian and English.

Adding POS tags and universal features to neural architectures of either type only marginally increases the classification accuracy but may even decrease it in some cases. As the accuracy does not increase enough with the best set of hyperparameters, we do not further discuss the effect of different pooling types on the performance; however, we can report that none of the pooling approaches consistently performed best.

2.4.4 Quality of morphological information

In the first ablation study, we evaluate the impact of the quality of morphological information. We replace the high quality (human-annotated) POS tags and morphological features used in Section 2.2.2 with those predicted by machine learning models. In this way, we test a realistic setting where the morphological information is at least to a certain degree noisy. We obtain POS tags and morphological features from Stanza models prepared for the involved languages [Qi et al., 2020]. To avoid overly optimistic results, we make sure to use models that are not trained on the same datasets used in our DP experiments. This is possible for a subset of five languages. Table 5 shows the results of DP models, trained with predicted morphological features. The first two lines of the table show the quality of used features, expressed as the proportion of tokens, for which POS tags and *all* morphological features are correctly predicted, i.e. we report their accuracy.

Table 4: Classification accuracies for baseline and enhanced models on the NER task in different languages. The upper part of the table shows LSTM models, and the lower part shows BERT models. The best scores for each language are marked with the bold typeface separately for LSTM and BERT models. We report the mean and standard deviation over five random splits of the data. The differences between the models of the same type and language are marginal.

	CRO	ENG
Model	Hate speech	Toxic comments
fastText + LSTM	$\mathbf{77.69 \pm 0.47}\%$	$88.14 \pm 0.12\%$
fastText + LSTM + uPOS	$76.63 \pm 0.77\%$	$88.16 \pm 0.15\%$
fastText + LSTM + uPOS + feats	$75.64 \pm 1.21\%$	$86.90 \pm 1.20\%$
fastText + LSTM + feats	$76.52 \pm 0.99\%$	$\mathbf{88.25 \pm 0.27}\%$
mBERT	$86.20 \pm 0.20\%$	$92.23 \pm 0.12\%$
mBERT + uPOS	$85.05 \pm 0.49\%$	$\mathbf{92.33 \pm 0.11}\%$
mBERT + uPOS + feats	$\mathbf{86.34 \pm 0.28}\%$	$92.29 \pm 0.20\%$
mBERT + feats	$86.22 \pm 0.35\%$	$92.20 \pm 0.13\%$

The general trend is that using predicted features results in much smaller (best case) performance increases, though some languages still see significant increases. For LSTM models, the increases range from 0.42% (English) to 1.60% (Slovene) for UAS, and from 0.67% (English) to 2.41% (Finnish) for LAS. For BERT models, the increases are marginal and range from 0.00% (English) to 0.49% (Finnish) for UAS, and from -0.10% (i.e. decrease, Slovene) to 0.66% (Finnish) for LAS. These results are consistent with the results of our NER experiments in section 2.4.1, where we have no access to human-annotated features and find that noisy features only help LSTM-based models.

As noisy features might require different training, we also tried to increase the maximal number of training steps to 15 epochs and repeated the tuning of hyperparameters for one language (Estonian). None of these changes improved the scores by a practically relevant margin.

These results indicate that adding predicted morphological features to models with BERT embeddings might not be practically useful since their quality needs to be very high. However, since human annotated morphological features improve the performance on the DP task, this suggests that there is a room for improvement in BERT pre-training. It seems that pre-training tasks of BERT (masked language modelling and next sentence prediction) do not fully capture the morphological information present in the language.

Table 5: UAS/LAS scores achieved by models that are trained with predicted (noisy) instead of human-annotated morphological features. In the first two lines, we report the proportion of tokens for which UPOS tags and UPOS features are correctly predicted.

	ENG	**EST**	**FIN**	**SLO**	**RUS**
UPOS accuracy (%):	92.45	91.15	87.59	80.45	89.32
feats accuracy (%):	93.92	88.86	86.20	78.74	85.22
baseline+LSTM	84.87/80.69	81.92/76.24	84.13/78.60	86.92/83.03	85.02/79.24
baseline+LSTM+UPOS	85.18/**81.36**	**82.44**/77.15	85.33/80.61	88.14/84.06	84.90/79.53
baseline+LSTM+UPOS+f.	85.04/81.02	82.24/**77.39**	**85.56**/**81.01**	**88.52**/**85.00**	**85.82**/**80.89**
baseline+LSTM+feats	**85.29**/81.22	81.42/76.19	84.89/80.43	88.12/84.12	84.74/79.56
baseline+mBERT	**91.43**/88.09	88.02/84.30	89.99/86.07	93.66/**91.67**	90.32/85.99
baseline+mBERT+UPOS	**91.43**/**88.33**	**88.20**/**84.69**	90.05/86.33	93.38/91.42	**90.59**/**86.17**
baseline+mBERT+UPOS+f.	91.08/88.05	88.06/84.39	**90.48**/**86.73**	**93.73**/91.57	90.34/85.74
baseline+mBERT+feats	90.90/87.74	87.88/84.42	89.77/85.91	93.37/91.23	89.65/85.27

2.4.5 Specific BERT models

In the second ablation study, we replace the embeddings obtained from the multilingual uncased BERT model with those obtained from more specific multilingual BERT models and monolingual BERT models. In experiments involving multilingual BERT models, we use Croatian/Slovene/English CroSloEngual BERT and Finnish/Estonian/English FinEst BERT [Ulčar and Robnik-Šikonja, 2020]. In experiments with monolingual BERT models, we use English bert-base-uncased [Devlin et al., 2019], Finnish bert-base-finnish-cased-v1 [Virtanen et al., 2019], and Russian RuBert [Kuratov and Arkhipov, 2019]. We only perform the experiments for a subset of studied languages for which we were able to find more specific BERT models. The aim is to check if the additional morphological features improve the performance of more language-specific BERT models. These are trained on a lower number of languages and larger amounts of texts in the involved languages compared to the original multilingual BERT model Devlin et al. [2019]. Due to this language-specific training, we expect these BERT models to better capture the languages' nuances, thus benefiting less from additional morphological features. The results are shown in Table 6 for trilingual BERT models and in Table 7 for monolingual BERT models.

Table 6: UAS/LAS scores achieved by trilingual BERT models, each trained on a smaller set of languages (3) than the original multilingual BERT model (104).

BERT variant	gual BERT			FinEst BERT	
Model/Language	CRO	ENG	SLO	FIN	EST
baseline+BERT	92.63/87.84	91.69/88.37	95.48/93.98	92.61/89.35	89.84/86.51
baseline+BERT+UPOS	**93.29**/**89.01**	**92.05**/**89.69**	95.72/94.27	**93.39**/**90.66**	**91.06**/88.67
baseline+BERT+UPOS+f.	92.95/88.33	91.99/89.65	**95.82**/**94.52**	93.23/90.61	91.01/**88.74**
baseline+BERT+feats	92.87/87.98	91.27/88.58	95.33/94.05	93.12/90.54	89.94/87.44

Table 7: UAS/LAS achieved by monolingual BERT models.

BERT variant	bert-base-uncased	bert-base-finnish-cased-v1	RuBert
Model	ENG	FIN	RUS
baseline + BERT	91.82/88.74	94.20/91.51	90.83/86.39
baseline + BERT + UPOS	**91.92**/89.60	**94.53**/**92.19**	**91.24**/**87.31**
baseline + BERT + UPOS + f.	91.81/**89.62**	94.08/91.73	91.04/87.04
baseline + BERT + feats	91.23/88.76	94.09/91.80	90.88/86.61

In most cases, the specific multilingual BERT models without additional features do as well as or better than the best performing original multilingual BERT model with additional features. The only worse LAS scores are achieved in English and Estonian, indicating that trilingual BERT models are better adapted to the task than the original multilingual models. The addition of morphological features increases the UAS and LAS even further. For UAS, the improvements are generally smaller than before (see Figure 2), with the only improvement larger than 1% being on Estonian. For LAS, the improvements are larger than UAS and range from 0.54% (Slovene) to 2.23% (Estonian); only one language sees an improvement of less than 1%. These results indicate that the additional morphological features still contain valuable information for the DP task, which the BERT models do not capture. We hypothesise that the slightly lower performance of FinEst BERT (without additions) on Estonian and the slightly larger increase when adding morphological features might be due to variability in the training process.

For monolingual models (the most language-specific), the results are mixed. Still, the general trend is that the additional features help little or not at all. The English monolingual model without and with morphological additions performs comparably to its multilingual (CroSloEngual) counterpart, and we do not see any additional increase in the performance. The Russian monolingual model achieves comparable UAS and LAS to the best performing original multilingual BERT model with additional features. With the addition of morphological features, the scores increase, but only marginally (under 0.5%). The Finnish monolingual model outperforms the best performing multilingual (FinEst) counterpart. The addition of

morphological features to the monolingual Finnish model increases the scores further: UAS by 0.33% and LAS by 0.68%, but these increases are not statistically significant.

2.5 Discussion of explicit morphological features

The results indicate that adding morphological information to CF prediction models is only marginally beneficial, but it can improve the performance in the NER and DP tasks. For the DP task, the improvement depends on the quality of the morphological features and the choice of the architecture. The additional morphological features consistently benefited LSTM-based models for NER and DP, both manually assigned (high quality) and predicted (noisy). We attribute the marginal improvements in the classification performance of the CF task to the nature of this task. In CF, semantics and word choice play a much more important role than exact word forms expressed with morphology. Therefore, the explicit morphological information is less important and might be of lower quality in the informal language, often appearing in this task.

For BERT-based models, the *predicted* features do not make any practical difference for the NER and DP tasks but manually assigned high-quality features improve the performance in the DP task. Testing different variants of BERT shows that language specialised variants, i.e. CroSloEngual BERT and FinEst BERT [Ulčar and Robnik-Šikonja, 2020], improve the performance on the DP task. The additional morphological information becomes less beneficial with a shift from multilingual towards monolingual models.

The comparison of different BERT variants indicates that BERT models do not completely capture the language morphology. Since BERT's release, several new pre-training objectives have been proposed, such as syntactic and semantic phrase masking [Zhou et al., 2020] and span masking [Joshi et al., 2020]. In further work, it makes sense to apply these models to the DP task to test how well they capture the morphology. Further, the effect of morphological features could be analysed on additional tasks and languages since the explicit morphological information does not seem to benefit them equally.

The work presented in Section 2 is described in full in [Klemen et al., 2020], attached as Appendix A.

3 Prediction Uncertainty Estimation

The information on the reliability of predictions is crucial in tasks that affect humans users, such as the ones appearing in WP3 (e.g., comment filtering, hate speech prediction, and other user generated contents), WP4 (e.g., text annotation related to news analysis), and T1.4 (explanation and visualisation of predictions). To address these needs, in this section, we describe a Bayesian method to obtain well-calibrated reliability estimates from transformer networks and test it on the hate speech detection problem in several languages. In Section 4, we show that a Bayesian ensemble of reliability estimates can also improve text annotation.

With the rise of social network popularity, hate speech phenomena have significantly increased [Davidson et al., 2017]. Hate speech not only harms both minority groups and the whole society, but it can lead to actual crimes [Bleich, 2011]. Thus, (automated) hate speech detection mechanisms are urgently needed. However, falsely accusing people of hate speech is also a problem. Many content providers rely on human moderators to reliably decide if a given text is offensive or not, but this is a mundane and stressful job that can even cause post-traumatic stress disorders[3]. There have been many attempts to automate the detection of hate speech in social media using machine learning. Still, existing models lack the quantification of reliability for their decisions.

In the last few years, recurrent neural networks (RNNs) were the most popular text classification choice. Long Short Term Memory (LSTM) networks, the most successful RNN architecture, were already successfully adapted to assess predictive reliability in hate speech classification [Miok et al., 2019a]. Re-

[3] https://www.bbc.com/news/technology-51245616

cently, neural network architecture with attention layers, called 'transformer architecture' [Vaswani et al., 2017], showed even better performance on almost all language processing tasks. Using transformer networks for masked language modelling produced breakthrough pretrained models, such as BERT [Devlin et al., 2019]. The attention mechanism, which is a crucial part of transformer networks, became an essential part of natural language understanding with a significant impact on language applications. We aim to investigate the behaviour of the attention mechanism concerning the reliability of predictions. We focus on the hate speech recognition task.

In hate speech detection, reliable predictions are needed to remove harmful content and possibly ban malicious users without harming the freedom of speech [Miok et al., 2019a]. Standard neural networks are inadequate for assessing predictive uncertainty, and the best solution is to use the Bayesian inference framework. However, classical Bayesian inference techniques do not scale well in neural networks with a high dimensional parameter space [Izmailov et al., 2020]. Various methods were proposed to overcome this problem [Myshkov and Julier, 2016]. One of the most efficient methods is Monte Carlo Dropout (MCD) [Gal and Ghahramani, 2016]. Its idea is to use dropout in neural networks as a regularisation technique [Srivastava et al., 2014] and interpret it as a Bayesian optimisation approach that takes samples from the approximate posterior distribution.

Our main contributions are:

1. We present a novel methodology for assessing prediction uncertainty in attention networks and BERT models.
2. Empirical analysis of the proposed Bayesian Attention Networks and MCD enhanced BERT models show improved calibration and prediction performance on hate speech detection tasks in several languages.

In the subsections, we describe the methodology for uncertainty assessment in transformer networks using attention layers and MCD. We shortly present the datasets, evaluation scenario, and the obtained results.

3.1 Bayesian Attention Networks

The BERT model [Devlin et al., 2019] is the transformer network that has achieved state-of-the-art results in many NLP tasks, including text classification [Xu et al., 2020, Gururangan et al., 2019, Chang et al., 2019]. We introduce Monte Carlo Dropout to transformer networks and BERT to construct their Bayesian variants. Analysis of different amounts of dropout, different variants of BERT modifications, and their hyper-parameters would require pretraining several different BERT models, which would require substantial computational resources. For example, pretraining a single BERT model on four TPUs requires more than a month of computational time [Ulčar and Robnik-Šikonja, 2020]. Thus, in this work, we explore two reliability extensions, i) the reliability on the encoder part of the BERT architecture trained from scratch (without pretraining) on the task of interest (in this work, referred to as the attention networks), and ii) reliability of pre-trained BERT models, using only fine-tuning. We believe this is a reasonable setting that sheds light on an important reliability aspect of transformer networks.

In Section 3.1.1, we first formally define the attention network architecture, and in Section 3.1.2, we make it Bayesian by introducing MCD. Finally, in Section 3.1.3, we describe how the MCD principle can be employed in already pre-trained BERT models.

3.1.1 Attention Networks

The basic architecture of the attention network follows the architecture of transformer networks [Vaswani et al., 2017] and is shown in Figure 4. The proposed architecture is similar to the encoder part of the transformer architecture. The difference is in the output part. A single output head was added to perform binary classification using the sigmoid activation function. The main difference to BERT, which

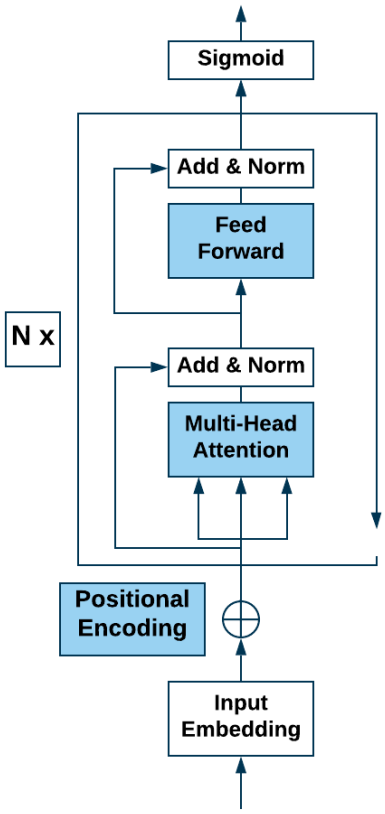


Figure 4: A scheme of Attention Networks. The dropout is introduced in the blue colored layers.

also uses just the encoder part of the transformer network, is that we do not use any pretraining. The second difference is that the attention network uses the classification head, and BERT has the language model head. In both cases, the output is composed of feed-forward layers followed by the non-linearity but with different dimensions in each case. By not relying on the pretraining, we are much more flexible concerning the number of layers and the number of neurons in each layer. For our tasks, we use orders of magnitude fewer parameters, e.g., we used a maximum of 3 million parameters (at the expense of losing information from pretraining).The architecture can contain many attention heads, where a single attention head is computed as:

$$o_h = \text{softmax}(\frac{\mathbf{Q} \cdot \mathbf{K}^T}{\sqrt{d_k}}) \cdot \mathbf{V}, \tag{1}$$

The attention matrices are commonly known as the query $\mathbf{Q}$, the key $\mathbf{K}$, and the value matrix $\mathbf{V}$. The normalizing factor, d_k, denotes the dimensionality of keys. The attention function can be described as mapping the query and the set of key-value pairs to the output. The query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values. The weight assigned to each value is computed by a compatibility function of the query with the corresponding key. Intuitively, the multiplication of query and key vectors with subsequent values can be understood as the extraction of *relations*. The softmax activation enables each pair of considered input tokens to be represented with a single real value. It effectively introduces *sparseness* into the weight space – only certain token pairs emerge with high weights and are relevant for the remaining part of the considered neural network architecture. In practice, multiple such heads can be concatenated and fed into the succeeding feed-forward layer. The application of softmax has been shown to emphasize only particular parts of the parameter space, thereby making the neural network more focused.

The positional encoding, as discussed by Vaswani et al. [2017], represents a matrix that encodes indi-

vidual positions in a matrix of the same dimensionality as the one holding the information on sequences (input embedding). The positional encoding was introduced to account for *word order*. Here, relative distances between different tokens are taken into account by incorporating the position-related signal into a given token representation.

While there are, in principle, many different ways of how attention networks can be extended with the Bayesian approach, we propose to use the well-established Monte Carlo Dropout.

3.1.2 Monte Carlo Dropout for Attention Networks

In our proposal, called 'Bayesian Attention Networks' (BAN), we use MCD within attention networks. Contrary to the original dropout setting, the dropout layers are also active during the prediction phase. In this way, the predictions are not deterministic. Instead, they are sampled from the *learned* distribution, thereby forming an ensemble of predictions. The obtained distribution can be, for example, inspected for higher moment properties and can offer additional information on the uncertainty of a given prediction. During the prediction phase, the dropout layers are activated again, and the output of a proportion of randomly selected neurons in those layers is set to zero. A forward pass on such partially activated architecture is repeated for a fixed number of samples, every time dropping different randomly selected neurons. The results of different passes can be combined to obtain the final prediction or further inspected as a probability distribution.

3.1.3 Monte Carlo Dropout for BERT

Monte Carlo dropout was used in the BERT model in the same way as in BAN. MCD can provide multiple predictions of a neural network during the test time, as long as the dropout was used during the training phase [Gal, 2016]. Training of neural networks with the dropout distributes the captured information across the network. During the prediction, such a trained neural network is robust. Using the dropout principle, a new prediction is possible in each forward pass. A sufficiently large set of such predictions can be used to estimate the prediction reliability. The BERT model is trained with 10% of dropout in all of the layers by default, thus allowing for multiple predictions using the described principle. We call this model 'MCD BERT'. A limitation of this approach is that a single dropout rate of 10% is used during training, while other dropout probabilities might be more suitable for reliability estimation. We leave this analysis for further work.

3.2 Datasets and implementation

To test the proposed methodology in the multilingual context, we used hate speech datasets in three languages, English, Croatian, and Slovene.

1. The **English** dataset[4] is extracted from hate speech and offensive language detection study of Davidson et al. [2017]. The subset of data we used consists of 5,000 tweets. We took 1,430 tweets labeled as hate speech and randomly sampled 3,670 tweets from the remaining 23,353 tweets.
2. The **Croatian** dataset was provided by the Styria media company within the EU project EMBEDDIA[5]. The texts consists of user comments on the news portal Večernji list[6]. The original dataset consists of 9,646,634 comments from which we selected 8,422 comments. 50% of instances were labeled as hate speech by human moderators. The other half was chosen randomly from non-problematic comments.

[4] https://github.com/t-davidson/hate-speech-and-offensive-language
[5] http://embeddia.eu
[6] https://www.vecernji.hr

3. The **Slovene** dataset was produced in the Slovenian national project FRENK[7]. The text dataset used in the experiment is a combination of two different studies of Facebook comments [Ljubešić et al., 2019]. The first group of comments was collected on LGBT homophobia topics, while the second on anti-migrants posts. In our final dataset, we used all of the 2,182 hate speech comments, and the same number of non-hate speech comments were randomly sampled.

We used three types of neural network architectures. As a baseline, we used MCD LSTM networks [Miok et al., 2019a], which include reliability information obtained with MCD. We compared that model with the newly proposed BAN and MCD BERT.

MCD LSTM networks consist of an embedding layer, an LSTM layer, and a fully connected layer within the word2Vec and ELMo embeddings. To obtain the best architectures for the LSTM and MCD LSTM models, we tested different numbers of units, batch sizes, dropout rates, etc.

For BERT, we used the BERT base model in English and the multilingual BERT variant for Croatian and Slovene. We used the HuggingFace implementation[8].

3.3 Results

We compare the predictive performance of four neural network architectures in Table 8. MCD LSTM and BERT serve as the baselines for comparison with the proposed BAN and MCD BERT. The MCD BERT model provides the best results for all three languages. BERT models are pre-trained on large amounts of text, which makes a significant difference compared to LSTM and BAN. MCD BERT is slightly better than BERT due to its better performance for the instances where BERT is uncertain. Here, multiple predictions reduce the prediction variance. MCD LSTM is more stable than BAN (see the standard deviation of F_1 scores in Table 8). We attribute this to the larger number of parameters in BAN and an insufficient number of training instances. BERT and MCD BERT models compensate for this problem with large scale pretraining.

Table 8: Predictive performance of compared models. We present the average classification accuracy and F_1 score with their standard deviations (in brackets), computed using 5-fold cross-validation. The best accuracy for each language is typeset in bold.

	English Tweets		Croatian Comments		Slovene Comments	
Model	Accuracy	F1	Accuracy	F1	Accuracy	F1
MCD LSTM	81.0 [1.2]	81.9 [1.3]	63.7 [1.0]	51.0 [3.3]	55.3 [0.69]	43.13 [0.8]
BAN	83.3 [1.7]	81.6 [3.4]	61.4 [2.0]	38.1 [8.6]	57.4 [1.7]	35.1 [6.3]
BERT	90.9 [0.7]	90.0 [0.7]	70.8 [1.0]	61.2 [1.5]	66.4 [5.0]	67.8 [2.5]
MCD BERT	91.4 [0.7]	90.4 [0.8]	71.5 [1.2]	62.9 [1.7]	68.4 [1.9]	68.6 [1.6]

The work presented in Section 3 is described in full in the journal paper of Miok et al. [2021], attached here as Appendix B. The initial version was presented in the workshop [Miok et al., 2020b] and is attached as Appendix C.

4 Bayesian Methods for Semi-supervised Text Annotation

Recent successful artificial intelligence applications in various fields, including natural language processing, are often due to long hours of human annotation when preparing datasets for machine learning. The annotation process transfers human knowledge to machine learning models. It is often done

[7] http://nl.ijs.si/frenk/ (Research on Inappropriate Electronic Communication)
[8] https://huggingface.co/transformers/model_doc/bert.html

under time pressure and with inadequate instructions or with insufficiently trained annotators. Aiming to make the annotation process easier, we study the possibility of designing a data labelling process that requires less human supervision. Our work is based on the reliability scores provided by the MCD BERT approach, described in Section 3. The work is relevant for text annotation and reannotation campaigns conducted in WP3 and WP4.

A fairly standard procedure in annotation quality control is to recheck the wrongly classified labels by using several prediction models. As an alternative, Bayesian inference, presented in Section 3, produces a distribution of possible decisions and can improve the selection of instances requiring reannotation [Miok et al., 2021]. Ensemble methods produce robust models that frequently provide significantly better predictions than individual models. The key strength of ensembles is that they can overcome the errors and shortcomings of individual ensemble members. A recently published ensemble method Multivariate Normal Mixture Conditional Likelihood Model (MM) [Pirš and Štrumbelj, 2019] tries to understand the predictors on the distributional level and use Bayesian inference to combine them. We evaluate MM's performance when combining multiple MCD BERT predictions on the hate speech detection task. We show that our methodology can serve as a helpful tool in the data annotation process.

This section proposes methods that can save time and resources during the text annotation process and improve prediction performance. As a test domain, we use hate speech detection in tweets, news comments, and Facebook comments. We investigate two performance-improving techniques, which can be summarized as our main contributions as follows.

1. We remove instances with uncertain classifications from the training set and show that fine-tuning on the cleaned dataset improves the performance of the BERT model. Less certain classifications can be selected for reannotation.
2. We combine predictions of machine learning models using the MM probabilistic ensemble method. The approach is beneficial for predictive performance.

In Section 4.1, we present a Bayesian probabilistic ensemble as a tool to combine multiple MCD BERT outputs. In Section 4.2, we present our experiments and their results that are promising for further text annotations conducted in WP3 and WP4.

4.1 Bayesian Probabilistic Ensemble

To alleviate the drawbacks of individual classification models, we propose the use of MM [Pirš and Štrumbelj, 2019], a Bayesian ensemble method suitable for combining correlated probabilistic predictions. MM is an extension of IBCC [Kim and Ghahramani, 2012], which combines non-probabilistic predictions. The method is based on finding the latent structure of combined predictions and provides new probabilities based on its distribution. Let m be the number of classes and r the number of individual models we are combining. The main idea is similar to Supra-Bayesian ensembles [Lindley, 1985], as we first transform individual probabilistic predictions with the inverse logistic transformation (log-odds) to move from [0,1] space to the $\mathbb{R}$ space. We merge the transformed predictions of individual models and get a $(m-1)r$-variate distribution. We model this latent distribution with multivariate normal mixtures, conditional on the true label in a similar fashion, as in linear discriminant analysis. Let θ represent estimated parameters and θ_t the subset of parameters estimated for observations with true label t. Let $T^* \in \{1, 2, ..., m\}$ be the response random variable for a new observation and $u^* \in \mathbb{R}^{(m-1)r}$ the transformed and merged predictions for this new observation. Probabilistic predictions for unseen data can then be generated by calculating the densities of merged predictions for new data:

$$p(T^* = t|u^*, \theta) = \frac{p(u^*|\theta_t)(\gamma_t n_t)}{\sum_{i=1}^{r} p(u^*|\theta_i)(\gamma_i n_i)},$$

where p is the multivariate normal mixture probability density, γ_t is the frequency prior for class t, and n_t is the number of true labels in class t in the training dataset. The method uses a regularization term, which increases the variance in any dimension that is difficult to model or has a detrimental effect on the

results, effectively decreasing its effect. For a complete Bayesian specification and the derivation of the Gibbs sampler, we refer the reader to [Pirš and Štrumbelj, 2019]. We used the same priors as proposed in this paper.

MM is well-suited for combining biased classifiers or classifiers with systematic errors. It can serve as a calibration tool for an individual classifier by learning its latent distribution. Since BERT is usually accurate but less well calibrated, the MM method can alleviate its miscalibration while improving or preserving its classification performance.

4.2 Evaluation and results

We first introduce the phases of our experiments, followed by the results. We use the same datasets as in Section 3.

We categorise classifications to trusted and untrusted based on the uncertainty measure from MCD BERT. In this way, we can detect borderline classifications that make a false impression of certainty. We remove the instances with uncertain classifications from the *training set* to improve the dataset on which the BERT model is fine-tuned. This provides better quality data for training and shall improve the quality of the resulting prediction model.

Using MCD BERT, we first obtain multiple predictions for each test set instance and compute their mean and variance. Using the mean, we determine the classification (hate speech or not). At the same time, the variance reports on the certainty of the BERT for this specific instance. Based on the variance, we group classifications into certain and uncertain. Unsurprisingly, removing the uncertain test set instances improves the prediction performance as shown in Table 9, but also leaves a portion of borderline instances unclassified.

Table 9: Performance of multilingual BERT model, after removing uncertain instances from the test set of 1000 comments.

Language	Metric	Full dataset	200 removed	500 removed	700 removed
EN	Accuracy	0.91	0.96	0.996	0.997
	Precision	0.90	0.95	0.992	0.994
	Recall	0.89	0.95	1	1
	F1	0.88	0.95	0.995	0.997
CRO	Accuracy	0.72	0.76	0.84	0.87
	Precision	0.68	0.71	0.80	0.85
	Recall	0.54	0.69	0.78	0.75
	F1	0.61	0.70	0.79	0.83
SLO	Accuracy	0.71	0.76	0.83	0.87
	Precision	0.60	0.65	0.70	0.65
	Recall	0.56	0.64	0.66	0.54
	F1	0.58	0.65	0.68	0.59

From Table 9, we can conclude that the variance of MCD BERT predictions is correlated with the performance of models: the more variance there is in the predictions, the less accurate the model. Thus, removing the uncertain classifications can seemingly improve the performance of the test set. A practical benefit of this is that uncertain classification could be passed back to annotators to recheck them.

While removing uncertain instances from the test set might sweep the problematic instances under the carpet, a more practical benefit is to use the uncertainty information to create a better training set. The test tweets/comments were removed based on how variate their predictions are. Thus, we repeatedly train the MCD BERT model on the part of the dataset and use it to obtain multiple predictions on the other part of the training dataset. In such a way, we collect multiple predictions for all original training tweets or comments and remove observations with the highest prediction variance. As a result of this

procedure, 15 and 18 percent of the most uncertain predictions were removed for the English and Slovene dataset, respectively. Croatian dataset contains many comments with high variability in their predictions; therefore, we removed around 35% of the most uncertain comments for this dataset.

Using prediction certainty to remove the uncertain instances from the training can improve the fine-tuning of BERT. The prediction results for Croatian and Slovenian datasets are improved while for the English dataset, this is not the case. We explain this by the fact that the English dataset is well-annotated with high-quality predictions. On the other hand, we believe that the Croatian and Slovenian datasets are less clean and contain several questionable annotations. This can be confirmed for the Croatian dataset created within the EMBEDDIA project, so we are well-informed about the annotation process.

We propose a Bayesian ensemble as a support method for the annotation process. As annotators can be distracted, biased, or influenced, we propose to use the MM method to provide them a hint of how shall they annotate the instances. From Table 10, we can observe that by combining probabilistic predictions of BERT, random forest, and support vector machines, we can further improve the predictive performance. The MM ensemble not only improves BERT's results but also provides better-calibrated predictions, as discussed in [Miok et al., 2021,0].

A common problem of text annotation campaigns is that annotators are not always sure about correct labels due to uncertainty in the text [Vincze, 2015, Szarvas et al., 2008]. On difficult texts, annotators frequently give ambiguous labels, and their annotations can be biased. Instead of asking annotators to label the raw text, it would be easier for them if they were warned ahead of difficult cases and given more time for them based on the probabilistic scores from an ensemble of predictive models. In future work, continuing in the context of WP3 and WP4, we will apply the suggested approach to assess the reliability of the annotation process in an actual interactive setting. We also plan to use the proposed uncertainty estimates in an active learning setting as a sampling method.

Table 10: The F_1 score of the hate speech classifiers and their ensemble.

Method	English	Croatian	Slovene
BERT	0.91	0.72	0.71
RF	0.83	0.67	0.65
SVM	0.86	0.71	0.69
MM	0.92	0.74	0.72

The work presented in Section 4 is described in full in [Miok et al., 2020a], attached here as Appendix D.

5 Unified Approach to Propositionalization and Embeddings

Text processing with embeddings results in a dense vector representation of texts. In case of documents, this data transformation results in the condensed tabular representation of documents, which is then used as an input to a learning algorithm of choice. A broader data transformation task of interest is when texts are mixed with other data modalities, which is referred to as heterogeneous data. Heterogeneous data may involve texts, relational databases as well as images, where the challenge is to transform all this data variety into a joint representation format.

Data preprocessing for machine learning is still a great challenge for a data scientist faced with large quantities of data in different forms and sizes. Most modern data processing techniques enable data fusion from different data types and formats into a single table data representation, which is expected by standard machine learning techniques including rule learning, decision tree learning, support vector machines (SVMs), and deep neural networks (DNNs), etc. The key element of modern data transformation methods' success is that similarities of original instances and their relations are encoded as

distances in the target vector space.

Two of the most prominent data transformation approaches are propositionalization and embeddings. While propositionalization [Kramer et al., 2001, Železný and Lavrač, 2006] is a well-known data transformation technique used in relational learning (RL) and inductive logic programming (ILP) [Muggleton, 1992, Lavrač and Džeroski, 1994, De Raedt, 2008], embeddings [Mikolov et al., 2013, Wu et al., 2018] have only recently been recognised by RL and ILP researchers as a powerful technique for preprocessing relational and complex structured data. In the relational learning context, both approaches take as input a relational data set (e.g., a given relational database) and transform it into a single data table format, which is then used as an input to a propositional learning algorithm of choice.

The main objective of task T1.3 of the EMBEDDIA project is to advance deep learning technology for texts in morphologically rich, less-resourced languages. However, future works will need to address heterogeneous data types even in text processing. A step in the direction of heterogeneous data transformations is to unify text embeddings with propositionalization and/or embeddings of relational data, allowing for generating unified outputs from the data of both modalities.

The main novelty of this section is a unifying methodology that combines propositionalization and embeddings, which benefits from the advantages of both in solving complex data transformation and learning tasks. The unifying methodology resulted in two new pipelines, PropDRM and PropStar, which implement an instance-based and a feature-based approach to data transformation and learning, respectively. Both approaches are computationally efficient and can successfully solve much larger tasks than the existing relational learning approaches. The work is relevant for further representation learning conducted within the project, e.g., in WP4, the modelling of relations between different topics and emotions could play an important role in the sentiment prediction.

The work outlined in Section 5 is described in full in [Lavrač et al., 2020], attached here as Appendix E.

6 Conclusions and further work

We presented several improvements to DNN technologies, developed in T1.3. The main results include adding explicit morphological information to LSTM and BERT neural network architectures, obtaining reliability estimates from DNNs, and unified representation learning for propositionalisation and embeddings.

The comparison of different BERT variants indicates that BERT models do not completely capture the language morphology. Since BERT's release, several new pre-training objectives have been proposed, such as syntactic and semantic phrase masking [Zhou et al., 2020] and span masking [Joshi et al., 2020]. In further work, it makes sense to apply these models to the DP task to test how well they capture the morphology. Further, the effect of morphological features could be analysed on additional tasks and languages since the explicit morphological information does not seem to benefit them equally.

The proposed reliability enhanced classifications could be used in many other domains, such as machine translation. Further, we aim to adapt other Bayesian approaches, such as SWAG, to transformer networks. We will further investigate how to apply MCD BERT to the annotation problem and how to remove uncertain instances from the training set. We will construct and test a workflow for semi-supervised text annotation in a real-world setting. Testing different dropout levels in the BERT model may provide a better understanding of its uncertainty and calibration.

It is worth investigating the integration of symbolic and deep learning, considering them as multitask learning approaches that try to integrate many different learning tasks and use embeddings as input representations.

7 Associated outputs

The work described in this deliverable has resulted in the following resources:

Description	URL	Availability
Morphological NER (fastText + LSTM)	github.com/EMBEDDIA/morphological-fasttext	Public (MIT)
Morphological NER (BERT)	github.com/EMBEDDIA/morphological-BERT	Public (Apache)
Morphological dependency parsing	github.com/EMBEDDIA/morphological-dependency-parsing	Public (MIT)
Morphological comment filtering	github.com/EMBEDDIA/morphological-comment-filtering	Public (MIT)
BAN code	github.com/EMBEDDIA/BAN	Public (MIT)
MCD BERT code	github.com/EMBEDDIA/Bayesian-BERT	Public (MIT)
Bayesian ensemble MM	github.com/EMBEDDIA/MM	Public (MIT)
PropStar and PropDRM code	github.com/EMBEDDIA/PropStar	Public (BSD 3)

Parts of this work are also described in detail in the following publications.

Citation	Status	Appendix
Matej Klemen, Luka Krsnik, and Marko Robnik-Šikonja. Enhancing deep neural networks with morphological information, 2020. ArXiv preprint http://arxiv.org/abs/2011.1243	Submitted	Appendix A
Kristian Miok, Blaž Škrlj, Daniela Zaharie, and Marko Robnik-Šikonja. To BAN or not to BAN: Bayesian attention networks for reliable hate speech detection. Cognitive Computation, 2021.	Accepted	Appendix B
Kristian Miok, Blaž Škrlj, Daniela Zaharie, and Marko Robnik-Šikonja. Bayesian BERT for Trustful Hate Speech Detection.Proceedings of the ICML 2020 Workshop on Uncertainty & Robustness in Deep Learning, 2020.	Published	Appendix C
Kristian Miok, Gregor Pirš, and Marko Robnik-Šikonja. Bayesian methods for semi-supervised text annotation. In Proceedings of the 14th Linguistic Annotation Workshop, COLING LAW 2020, 2020	Published	Appendix D
Nada Lavrač, Blaž Škrlj, and Marko Robnik-Šikonja. Propositionalization and embeddings: Two sides of the same coin. Machine Learning, 109:1465–1507, 2020.	Published	Appendix E

References

Erik Bleich. The rise of hate speech and hate crime laws in liberal democracies. *Journal of Ethnic and Migration Studies*, 37(6):917–934, 2011.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

Wei-Cheng Chang, Hsiang-Fu Yu, Kai Zhong, Yiming Yang, and Inderjit Dhillon. X-BERT: eXtreme multi-label text classification with BERT. *arXiv preprint arXiv:1905.02331*, 2019.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. Automated hate speech detection and the problem of offensive language. In *Proceedings of the 11th International AAAI Conference on Web and Social Media*, ICWSM '17, pages 512–515, 2017.

Luc De Raedt. *Logical and Relational Learning*. Springer, 2008.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, 2019.

Timothy Dozat and Christopher D. Manning. Deep biaffine attention for neural dependency parsing, 2016.

Daniel Edmiston. A systematic analysis of morphological content in BERT models for multiple languages. arXiv:2004.03032, 2020.

Yarin Gal. *Uncertainty in deep learning*. PhD thesis, University of Cambridge, 2016.

Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, pages 1050–1059, 2016.

Suchin Gururangan, Tam Dang, Dallas Card, and Noah A. Smith. Variational pretraining for semi-supervised text classification. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5880–5894, July 2019.

Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8): 1735–1780, November 1997.

Pavel Izmailov, Wesley J Maddox, Polina Kirichenko, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Subspace inference for bayesian deep learning. In *Uncertainty in Artificial Intelligence*, pages 1169–1179, 2020.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. SpanBERT: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77, 2020.

Daniel Jurafsky and James H. Martin. *Speech and Language Processing (2nd Edition)*. Prentice-Hall, Inc., 2009.

Gopal K Kanji. *100 statistical tests*. Sage, 2006.

Hyun-Chul Kim and Zoubin Ghahramani. Bayesian Classifier Combination. In *International Conference on Artificial Intelligence and Statistics*, pages 619–627, 2012.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. Character-aware neural language models. In *AAAI*, pages 2741–2749, 2016.

Matej Klemen, Luka Krsnik, and Marko Robnik-Šikonja. Enhancing deep neural networks with morphological information, 2020. URL `http://arxiv.org/abs/2011.12432`. Submitted.

Stefan Kramer, Nada Lavrač, and Peter Flach. Propositionalization approaches to relational data mining. In S. Džeroski and N. Lavrač, editors, *Relational Data Mining*, pages 262–291. Springer-Verlag, 2001.

Yuri Kuratov and Mikhail Arkhipov. Adaptation of deep bidirectional multilingual transformers for russian language. In *Computational Linguistics and Intellectual Technologies: Proceedings of the International Conference "Dialogue 2019"*, 2019.

Nada Lavrač and Sašo Džeroski. *Inductive Logic Programming: Techniques and Applications*. Ellis Horwood, 1994.

Nada Lavrač, Blaž Škrlj, and Marko Robnik-Šikonja. Propositionalization and embeddings: Two sides of the same coin. *Machine Learning*, 109:1465–1507, 2020.

Dennis V Lindley. Reconciliation of discrete probability distributions. *Bayesian statistics*, 2:375–390, 1985.

Nikola Ljubešić, Darja Fišer, and Tomaž Erjavec. The FRENK datasets of socially unacceptable discourse in Slovene and English. In *International Conference on Text, Speech, and Dialogue*, pages 103–114, 2019.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

Kristian Miok, Dong Nguyen-Doan, Blaž Škrlj, Daniela Zaharie, and Marko Robnik-Šikonja. Prediction uncertainty estimation for hate speech classification. In *International Conference on Statistical Language and Speech Processing*, pages 286–298, 2019a.

Kristian Miok, Gregor Pirš, and Marko Robnik-Šikonja. Bayesian methods for semi-supervised text annotation. In *Proceedings of the 14th Linguistic Annotation Workshop, COLING LAW 2020*, 2020a.

Kristian Miok, Blaž Škrlj, Daniela Zaharie, and Marko Robnik-Šikonja. Bayesian BERT for trustful hate speech detection. In *Proceedings of the ICML 2020 Workshop on Uncertainty & Robustness in Deep Learning*, 2020b.

Kristian Miok, Blaž Škrlj, Daniela Zaharie, and Marko Robnik-Šikonja. To BAN or not to BAN: Bayesian attention networks for reliable hate speech detection. *Cognitive Computation*, 2021. (accepted).

Stephen Muggleton, editor. *Inductive Logic Programming*. Academic Press Ltd., London, 1992.

Pavel Myshkov and Simon Julier. Posterior distribution analysis for Bayesian inference in neural networks. In *Workshop on Bayesian Deep Learning, NIPS*, 2016.

Joakim Nivre, Mitchell Abrams, Željko Agić, et al. Universal Dependencies 2.6, 2020. URL `http://hdl.handle.net/11234/1-2988`. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Telmo Pires, Eva Schlinger, and Dan Garrette. How multilingual is multilingual BERT? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4996–5001, 2019.

Gregor Pirš and Erik Štrumbelj. Bayesian combination of probabilistic classifiers using multivariate normal mixtures. *Journal Machine Learning Research*, 20:51–1, 2019.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. Stanza: A Python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 2020.

Radu Soricut and Franz Och. Unsupervised morphology induction using word embeddings. In *Proceedings of NAACL-HLT*, pages 1627–1637, 2015.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Milan Straka and Jana Straková. Tokenizing, POS tagging, lemmatizing and parsing UD 2.0 with UDPipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 88–99, 2017.

György Szarvas, Veronika Vincze, Richárd Farkas, and János Csirik. The bioscope corpus: annotation for negation, uncertainty and their scope in biomedical texts. In *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing*, pages 38–45, 2008.

Matej Ulčar and Marko Robnik-Šikonja. FinEst BERT and CroSloEngual BERT: less is more in multilingual models. In *Proceedings of Text, Speech, and Dialogue, TSD 2020*, pages 104–111, 2020.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

Veronika Vincze. *Uncertainty detection in natural language texts*. PhD thesis, University of Szeged, 2015.

Antti Virtanen, Jenna Kanerva, Rami Ilo, Jouni Luoma, Juhani Luotolahti, Tapio Salakoski, Filip Ginter, and Sampo Pyysalo. Multilingual is not enough: BERT for Finnish. arXiv 1912.07076, 2019.

Frank Wilcoxon, SK Katti, and Roberta A Wilcox. Critical values and probability levels for the Wilcoxon rank sum test and the Wilcoxon signed rank test. *Selected tables in mathematical statistics*, 1:171–259, 1970.

Ledell Yu Wu, Adam Fisch, Sumit Chopra, Keith Adams, Antoine Bordes, and Jason Weston. StarSpace: Embed all the things! In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, pages 5569–5577, 2018.

Yige Xu, Xipeng Qiu, Ligao Zhou, and Xuanjing Huang. Improving BERT fine-tuning via self-ensemble and self-distillation. *arXiv preprint arXiv:2002.10345*, 2020.

Filip Železný and Nada Lavrač. Propositionalization-based relational subgroup discovery with RSD. *Machine Learning*, 62(1-2):33–63, 2006.

Junru Zhou, Zhuosheng Zhang, Hai Zhao, and Shuailiang Zhang. LIMIT-BERT: Linguistic informed multi-task BERT. arXiv 1910.14296, 2020.

Appendix A: Enhancing Deep Neural Networks with Morphological Information

ENHANCING DEEP NEURAL NETWORKS WITH MORPHOLOGICAL INFORMATION

A PREPRINT

Matej Klemen, Luka Krsnik, Marko Robnik-Šikonja
University of Ljubljana, Faculty of Computer and Information Science
Večna pot 113, Ljubljana, Slovenia
mk3141@student.uni-lj.si, krsnik.luka92@gmail.com, marko.robnik@fri.uni-lj.si

November 26, 2020

ABSTRACT

Currently, deep learning approaches are superior in natural language processing due to their ability to extract informative features and patterns from languages. Two most successful neural architectures are LSTM and transformers, the latter mostly used in the form of large pretrained language models such as BERT. While cross-lingual approaches are on the rise, a vast majority of current natural language processing techniques is designed and applied to English, and less-resourced languages are lagging behind. In morphologically rich languages, plenty of information is conveyed through changes in morphology, e.g., through different prefixes and suffixes modifying stems of words. The existing neural approaches do not explicitly use the information on word morphology. We analyze the effect of adding morphological features to LSTM and BERT models. As a testbed, we use three tasks available in many less-resourced languages: named entity recognition (NER), dependency parsing (DP), and comment filtering (CF). We construct sensible baselines involving LSTM and BERT models, which we adjust by adding additional input in the form of part of speech (POS) tags and universal features. We compare the obtained models across subsets of eight languages. Our results suggest that adding morphological features has mixed effects depending on the quality of features and the task. The features improve the performance of LSTM-based models on the NER and DP tasks, while they do not benefit the performance on the CF task. For BERT-based models, the added morphological features only improve the performance on DP when they are of high quality (i.e. manually checked), while they do not show any practical improvement when they are predicted. As in NER and CF datasets manually checked features are not available, we only experiment with the predicted morphological features and find that they do not cause any practical improvement in performance.

Keywords Deep learning · Natural language processing · Morphologically rich languages

1 Introduction

The use of deep learning for processing natural language is becoming a standard, with excellent results in a diverse range of tasks. Two state-of-the-art architectures for text-related modeling are long short-term memory (LSTM) networks [Hochreiter and Schmidhuber, 1997] and transformers [Vaswani et al., 2017]. LSTMs are recurrent neural networks that process the text sequentially, meaning that they process text one token at a time, building up its internal representation in hidden states of the network. Due to the recurrent nature of LSTM, which degrades the efficiency of parallel processing, as well as demonstrated improvements in performance, models based on the transformer architecture are gradually replacing LSTMs across many tasks. Transformers can process the text in parallel, using self-attention and positional embeddings to model the sequential nature of the text.

A common trend in using transformers is to first pre-train them on large monolingual corpora with abstract, general-purpose objective, and then fine-tune them for a specific task, such as text classification. For example, the BERT

(Bidirectional Encoder Representations from Transformers) architecture Devlin et al. [2019] uses transformers and is pretrained with masked language modelling and order of sentences prediction tasks to build a general language understanding model. During the fine-tuning for a specific downstream task, additional layers are added to the BERT model, and the model is trained on specific data to capture the specific knowledge required to perform the task.

Most of the research in the natural language processing (NLP) area focuses on English, ignoring the fact that English is specific in terms of the low amount of information expressed through morphology (English is so-called analytical language). In our work, we focus on adapting modern deep neural networks, namely LSTMs and BERT, for several morphologically rich languages, by explicitly including the morphological information. The languages we analyze contain rich information about grammatical relations in the morphology of words instead of in particles or relative positions of words (as is the case in English). For comparison, we also evaluate our models on English. Although previous research has shown that the state of the art methods such as BERT already captures some information contained in the morphology [Pires et al., 2019, Edmiston, 2020], our experiments involve several languages with rich morphology where neural networks could benefit from explicit morphological features.

Specifically, we present methods which combine BERT with separately encoded morphological properties: universal part of speech tags (uPOS tags) and universal features (grammatical gender, tense, conjugation, declination, etc). We evaluate them on three downstream tasks: named-entity recognition (NER), dependency parsing (DP), and comment filtering (CF). We perform similar experiments on LSTM networks and compare the results for both architectures. Besides English, we analyze eight more languages: Croatian, Estonian, Finnish, Latvian, Lithuanian, Russian, Slovene and Swedish. The choice of these languages reflects a mix of different language groups (Balto-Slavic, Germanic, and Uralic), for which we were able to obtain sufficient resources (datasets, embeddings, and corpora), due to their coverage in the EU EMBEDDIA project[1].

Our experiments show that the addition of morphological features has mixed effects depending on the task. Across the tasks where the added morphological features improve the performance, we show that (1.) they benefit the LSTM-based models even if the features are noisy and (2.) they benefit the BERT-based models only when the features are of high quality (i.e. human checked), suggesting that BERT models already capture the morphology of the language; however, there is a room for improvement either in designing pre-training objectives that can capture these properties or when high-quality features are available.

The remainder of this paper is structured as follows. In Section 2, we present different attempts to use morphological information in machine learning, in particular neural networks, as well as an overview of recent work in the three evaluation tasks. In Section 3, we describe the used datasets and their properties. In Section 4, we present the baseline models and models with additional morphological information, whose performance we discuss in Section 5. Finally, we summarize our work and present directions for further research in Section 6.

2 Related work

We shortly review the related work on the use of morphological information within neural networks. We split the works based on three evaluation tasks: NER, DP, and CF.

2.1 Morphological features in NER

Recent advances in NER are mostly based on deep neural networks. One of the first usages of deep neural networks in NER is the work by Collobert and Weston [2008], who propose an architecture that is jointly trained on six different tasks, including NER, and show that this transfer learning approach generalizes better due to learning a joint representation for different tasks. A standard approach to NER is to use word [Huang et al., 2015] or character [Kuru et al., 2016] embeddings of the input, followed by one or more neural layers, mostly recurrent ones, such as LSTMs.

While sequence modelling on the character-level already has a potential to encode morphological information, several authors show that the performance of neural networks on the NER task can be improved by adding explicit information about morphological properties of the text. Güngör et al. [2017] show that embedding morphosyntactic properties improves NER performance on the Turkish language, while Simeonova et al. [2019] use part of speech embeddings to achieve improved performance on the NER task in the Bulgarian language. Our work is similar to these approaches in the sense that we add different morphological features to the neural networks. However, we experiment on a larger scale, using more languages from different language groups, different evaluation tasks, and with modern neural architectures (BERT), taking the LSTM networks as the baseline.

[1] http://embeddia.eu

2.2 Morphological features in dependency parsing

Similarly to NER, recent progress in DP is dominated by neural approaches, which can be roughly divided into two categories, transition-based [Yamada and Matsumoto, 2003, Nivre, 2003] and graph-based approaches [McDonald et al., 2005]. Some works do not fall into either category, e.g., DP is treated as a sequence-to-sequence task by Li et al. [2018]. The two categories differ in how dependency trees are produced from the output of prediction models. In the transition-based approach, a model is trained to predict a sequence of parsing actions which produce a valid dependency tree, while in the graph-based approach, a model is used to score candidate dependency trees via the sum of scores of their substructures (e.g., arcs). One of the earlier successful approaches to neural DP was presented by Chen and Manning [2014], who replaced the commonly used sparse features with dense embeddings of words, part of speech tags and arc labels, in combination with the transition-based parser. This approach improved both the accuracy and parse speed. Pei et al. [2015] introduced a similar approach to graph-based parsers. Later approaches improve upon the earlier methods by automatically extracting more information that guides the parsing. E.g., researchers use LSTM networks to inject context into local embeddings [Kiperwasser and Goldberg, 2016], apply contextual word embeddings [Kulmizev et al., 2019], or train graph neural networks [Ji et al., 2019].

The use of morphological features in DP is quite common. For example, Chen and Manning [2014] note that POS tag embeddings strongly contribute to the performance of their system in English and Chinese. Morphological features are commonly used with morphologically rich languages, e.g., Arabic [Marton et al., 2010], German [Seeker and Kuhn, 2011], Lithuanian [Kapočiūtė-Dzikienė et al., 2013], or Persian language [Khallash et al., 2013], with the consensus that they improve the accuracy of parsing. Concerning adding morphological properties to BERT, Kondratyuk and Straka [2019] train the multilingual BERT model for multiple tasks, including POS and morphological tagging, and achieve improved results across multiple languages. This approach uses transfer learning to implicitly injects the morphological information into BERT models, while our approach does that explicitly, through morphological features on the input.

2.3 Morphological features in comment filtering

The literature for the task of NER covers multiple related tasks, such as hate speech, offensive speech, political trolling, detecting commercialism, etc. Recent approaches involve variants of deep neural networks, though standard machine learning approaches are still popular as shown in the survey of Fortuna and Nunes [2018]. These approaches typically use features such as character n-grams, word n-grams, and sentiment of the sequence. Two examples are the works of Malmasi and Zampieri [2017], who classify hate speech in English tweets, and Van Hee et al. [2015], who classify different levels of cyberbullying in Dutch posts on ask.fm. Scheffler et al. [2018] combine word embeddings with the above-mentioned features to classify German tweets. They observe that combining both n-gram features and word embeddings brings only a small improvement over only using only one of them. The effectiveness of using features describing syntactic dependencies for toxic comments classification on English Wikipedia comments is shown by Shtovba et al. [2019].

Neural architectures used include convolutional neural networks [Georgakopoulos et al., 2018] and LSTM networks [Gao and Huang, 2017, Miok et al., 2019], typically improving the performance over standard machine learning approaches. The NER topic has also been the focus of shared tasks on identification and categorization of offensive language [Zampieri et al., 2019] and multilingual offensive language identification [Zampieri et al., 2020]. The reports of these tasks show the prevalence and general success of large pretrained contextual models such as BERT, though, surprisingly, the best performing model for the subtask B of SemEval-2019 Task 6 was rule-based [Han et al., 2019].

2.4 Linguistic knowledge combined with neural networks

Large pretrained models such as BERT show superior performance across many tasks. Due to a lack of theoretical understanding of this success, many authors study how and to what extent BERT models can capture various information, including different linguistic properties. An overview of recent studies in this area sometimes referred to as BERTology, is compiled by Rogers et al. [2020]. Two common approaches to study BERT are i) add additional properties to BERT models and observe the difference in performance on downstream tasks, ii) a technique called probing [Conneau et al., 2018], where the BERT model is trained (fine-tuned) to predict a studied property. For example, Jawahar et al. [2019] investigate what type of information is learned in different layers of the BERT English model and find that it captures surface features in lower layers, syntactic features in middle layers, and semantic features in higher layers. Similarly, Lin et al. [2019] find that BERT encodes positional information about tokens in lower layers and then builds increasingly abstract hierarchical features in higher layers. Tenney et al. [2019] use probing to quantify where different types of linguistic properties are stored inside BERT's architecture and suggest that BERT implicitly learns the steps performed in classical (non-end-to-end) natural language processing pipeline. However, Elazar et al. [2020] point out possible flaws in the probing technique, suggesting amnesic probing as an alternative. They arrive at slightly

different conclusions about BERT layer importance; for example, they show that POS information affects the predictive performance much more in upper layers.

Specifically for morphological properties, Zhou et al. [2020a] study whether the inclusion of POS information is still necessary for a neural dependency parser. Somewhat similarly to the previously mentioned transfer learning approach of Kondratyuk and Straka [2019], they add POS information to the DP task in English and Chinese by training a model with multi-task learning objective; they conclude that POS tags are still beneficial in a neural dependency parser, which our experiments on a larger set of languages confirm. In contrast to our work, which studies the impact of morphological features in BERT on the performance in downstream tasks, Edmiston [2020] studies the morphological content present in BERT. The author uses probing, i.e. he trains a linear classifier to predict morphological features based on features of a certain hidden layer. Based on the achieved high performance, he argues that BERT models capture much morphological information and partition their embedding space into linearly-separable regions, correlated with morphological properties.

3 Data

In this section, we describe the datasets used in our experiments, separately for each of the three tasks: NER, DP, and NER.

3.1 Named entity recognition

In the NER experiments, we use datasets in eight languages: Croatian, English, Estonian, Finnish, Latvian, Russian, Slovene and Swedish. We omit the Lithuanian language from this experiment, as models for obtaining POS tags and universal features were not available in the used tools at the time of performing the experiments. The number of sentences and tags present in the datasets is shown in Table 1. The label sets used in datasets for different languages vary, meaning that some contain more fine-grained labels than others. To make results across different languages consistent, we trim labels in all datasets to the four common ones: location (LOC), organization (ORG), person (PER), and "no entity" (OTHR).

Table 1: The collected datasets for NER task and their properties: the number of sentences and tagged words.

Language	Dataset	Sentences	Tags
Croatian	hr500k	24794	28902
English	CoNLL-2003 NER	20744	43979
Estonian	Estonian NER corpus	14287	20965
Finnish	FiNER data	14484	16833
Latvian	LV Tagger train data	9903	11599
Russian	factRuEval-2016	4907	9666
Slovene[2]	ssj500k	9489	9440
Swedish	Swedish NER	9369	7292

3.2 Dependency parsing

To test morphological neural networks on the DP task, we used datasets in nine languages (Croatian, English, Estonian, Finnish, Latvian, Lithuanian, Russian, Slovene and Swedish). The datasets are obtained from the Universal Dependencies [Nivre et al., 2020]. The number of sentences and tokens is shown in Table 2.

3.3 Comment filtering

While there exist comparable datasets across different languages for the NER and DP task, for the CF task, no such standard datasets exist. For that reason, in our experiments on CF, we used two languages with adequate datasets: English and Croatian.

For English experiments, we use a subset of toxic comments from Wikipedia's talk page edits[3]. The comments are annotated with six possible labels: toxic, severe toxic, obscene language, threats, insults, and identity hate (making

[2]The Slovene ssj500k originally contains more sentences, but only 9489 are annotated with named entities.

[3]https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/data

Table 2: Dependency parsing datasets and their properties: the treebank, number of sentences, number of tokens, and information about the size of the split.

Language	Treebank	Tokens	Sentences	Train	Validation	Test
Croatian	SET	199409	9010	6914	960	1136
English	EWT	254855	16622	12543	2002	2077
Estonian	EDT	438171	30972	24633	3125	3214
Finnish	TDT	202697	15135	12216	1364	1555
Latvian	LVTB	220536	13643	10156	1664	1823
Lithuanian	ALKSNIS	70051	3642	2341	617	684
Russian	GSD	98000	5030	3850	579	601
Slovene	SSJ	140670	8000	6478	734	788
Swedish	Talbanken	96858	6026	4303	504	1219

a total of six binary target variables). We extracted comments from three categories: toxic, severe toxic, threats, and identity hate; a total of $21,541$ instances. We randomly chose the same amount of comments that do not fall in any of the mentioned categories, obtaining the final dataset of $43,082$ instances.

For Croatian language experiments, we form a dataset from user comments published in the Croatian news site 24sata. The comments used are either unproblematic or labelled with one of the eight rules they break. Below we briefly describe the types of problematic content present in the database.

1. Advertising, content unrelated to the topic, spam, copyright infringement, citation of abusive comments or any other comments that are not allowed on the portal.
2. Direct threats to other users, journalists, admins or subjects of articles, which may also result in criminal prosecution.
3. Verbal abuse, derogation and verbal attack based on national, racial, sexual or religious affiliation, hate speech and incitement.
4. Collecting and publishing personal information, uploading, distributing or publishing pornographic, obscene, immoral or illegal content and using a vulgar or offensive nickname that contains the name and surname of others.
5. Publishing false information for deception or slander, and "trolling" - deliberately provoking other commentators.
6. Use of bad language, unless they are used as a stylistic expression, or are not addressed directly to someone.
7. Writing in other languages besides the Croatian, in other scripts besides Latin or writing with all caps.
8. Verbally abusing other users and their comments, article authors, and direct or indirect article subjects, calling the admins out or arguing with them in any way.

Unfortunately, this data is proprietary and is not publicly available. We extracted all the $17,868$ comments from the third category (the hate speech label) and approximately the same number of comments that do not break any rules. Our final dataset contains $35,635$ instances.

4 Neural networks with morphological features

In this section, we describe the architectures of neural networks used in our experiments. Their common property is that we enhance standard word embeddings based inputs with embeddings of morphological features. We work with recent successful neural network architectures, LSTMs and transformers, i.e. BERT models. A detailed description of architectures is available in the following subsections, separately for each of the three evaluation tasks. For each task and architecture, we describe the baseline architecture and the enhanced one.

4.1 Named entity recognition models

In the NER task, we use two baseline neural networks (LSTM and BERT), and the same two models with additional morphological information: POS tag embeddings and universal feature embeddings. The baseline models and their enhancements are displayed in Figure 1.

The first baseline model (left-hand side of Figure 1) is a unidirectional (left-to-right) LSTM model, which takes as an input a sequence of tokens, embedded using 300-dimensional fastText embeddings [Bojanowski et al., 2017] which are particularly suitable for morphologically rich languages as they work with subword inputs[4]. For each input token, its LSTM hidden state is extracted and passed through the linear layer to compute its NER score (probability for each of the four NER labels).

The second baseline model (right-hand side of Figure 1) is the multilingual BERT model (the base cased variant). In our experiments, we follow the sequence tagging approach suggested by the authors of BERT [Devlin et al., 2019]. Here, the input sequence is prepended with a special token [CLS] and passed through the BERT model. The output of the last BERT hidden layer is passed through the linear layer to obtain the predictions for NER labels.

Both baseline models (LSTM and BERT) are enhanced with the same morphological information: POS tag embeddings and universal feature embeddings for each of the input tokens. We embed the POS tags using 15-dimensional embeddings. For each of the 23 universal features used (we omitted the *Typo* feature, as the version of the POS tagger we used did not annotate this feature), we constructed 15-dimensional embeddings. We computed the POS tags and morphological features using the Stanza [Qi et al., 2020] system in the universal dependencies mode. In the enhanced architectures, we included another linear layer before the final linear classification layer to model possible interactions.

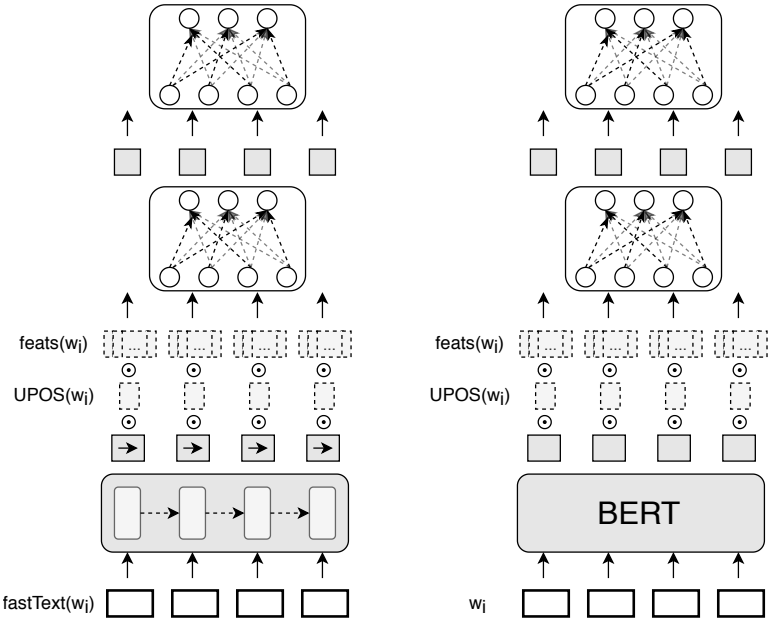


Figure 1: The baseline LSTM-based (left) and BERT-based (right) models for the NER task along with our modifications with morphological information. The dotted border of POS vectors, morphological features (feats), and the linear layer marks that their use is optional and varies across experiments. The $\odot$ symbol between layers represents the concatenation operation. The w_i symbol stands for token i; in case of LSTM, tokens enter the model sequentially and we show the unrolled network, while BERT processes all token simultaneously.

4.2 Dependency parsing models

As the baseline model in the DP task, we use the deep biaffine graph-based dependency parser [Dozat and Manning, 2016]. The enhancements with the morphological information are at the input level. The baseline model and its enhancements are shown in Figure 2.

The baseline parser combines a multi-layer bidirectional LSTM network with a biaffine attention mechanism to jointly optimize prediction of arcs and arc labels. We leave the majority of baseline architectural hyperparameters at values

[4]The precomputed embeddings are available at https://fasttext.cc/docs/en/crawl-vectors.html.

described in the original paper (3-layer bidirectional LSTM with 100-dimensional input word embeddings, and the hidden state size of 400).

In our experiments, we concatenate the non-contextual word embeddings with various types of additional information. The first additional input is contextual word embeddings, which we obtain either by using the hidden states of an additional single-layer unidirectional LSTM or by using a learned linear combination of all hidden states of BERT. Although the LSTM layers are already present in the baseline parser, we include an additional layer at the input level to keep the experimental settings similar across our three evaluation tasks. The second additional input is universal POS embeddings (UPOS), and the third one are universal feature embeddings (feats). These embeddings are concatenated separately for each token of the sentences. The size of the additional LSTM layer, POS tag embeddings and universal feature embedding are treated as tunable hyperparameters. As the baseline input embeddings, we use pre-trained 100-dimensional fastText embeddings, which we obtain by reducing the dimensionality of publicly available 300-dimensional vectors with fastText's built-in dimensionality reduction tool.

In DP experiments, we use POS tags and morphological features of two origins. The first source is human annotations provided in the used datasets. The second source of morphological information is predictions of Stanza models [Qi et al., 2020]. These two origins are used in the assessment of the quality of morphological information; namely, we check if manual human annotations provide any benefit compared to automatically determined POS tags and features.

4.3 Comment filtering models

In the NER evaluation task, we add additional morphological information to standard LSTM and BERT models. The baseline models and enhanced models for this task are similar to those in the NER evaluation, though we operate at the sequence level here as opposed to the token level in the NER task. The architecture of models is shown in Figure 3. As baselines, we take a single layer unidirectional LSTM network (the top part of Figure 3), and the multilingual base uncased BERT model (the bottom part of Figure 3). The difference in the used BERT dialect (*uncased* as opposed to the *cased* in the NER task) is due to better performance detected in preliminary experiments on a separate Croatian validation set.

In the LSTM baseline model, the words of the input sequence are embedded using pre-trained 300-dimensional fastText embeddings. As the representation of the whole sequence, we take the output of the last hidden state, which then passes through the linear layer to obtain the prediction scores. In the BERT baseline model, we take the sequence classification approach suggested by the authors of BERT. The input sequence is prepended with the special [CLS] token and passed through BERT. The sequence representation corresponds to the output of the last BERT hidden layer for the [CLS] token, which is passed through a linear layer to obtain the prediction scores.

As in other tasks, we augment the baseline models with POS tag and universal feature embeddings. We obtain the embeddings for each token separately from UDPipe models [Straka and Straková, 2017]. The reason for not using Stanza in this set of experiments is that we started experimenting before its release. Additional testing with Stanza did not show a difference in performance, so we stayed with the UDPipe models for this set of experiments. We combine the obtained embeddings using three different pooling mechanisms: mean, weighted combination, or LSTM pooling. Given the POS tag or universal feature embeddings, the mean pooling outputs the mean of all token embeddings; the weighted pooling outputs the weighted combination of token embeddings, and the LSTM pooling outputs the last hidden state obtained by passing the sequence of embeddings through the LSTM network. Both the embedding sizes and the type of pooling are treated as tunable hyperparameters. The coefficients of the weighted combination are learned by projecting the sequence embeddings into a sequence of independent dimension values which are normalized with the softmax. This compresses the embedding sequence, establishes its fixed length, and allows for different morphological properties to have a different impact in the sequence representation. This approach tests the contextual encoding of morphological properties. For example, adjectives might be assigned a higher weight than other POS tags due to their higher emotional contents that often indicates insults.

5 Evaluation

In this section, we first present the evaluation scenario for the three evaluation tasks, followed by results, presented separately for each of the tasks. We end the section with the ablation study, performed on the DP task. In the ablation study, we investigate the effect of additional morphological features in three situations where we tweak one aspect of training procedure at a time: (1) we increase the maximum training time of our models by additional 5 epochs, (2) we replace the human-annotated features with the ones automatically predicted by machine learning models, and (3) we replace the embeddings from general multilingual BERT with those from more specific multilingual and monolingual BERT models.

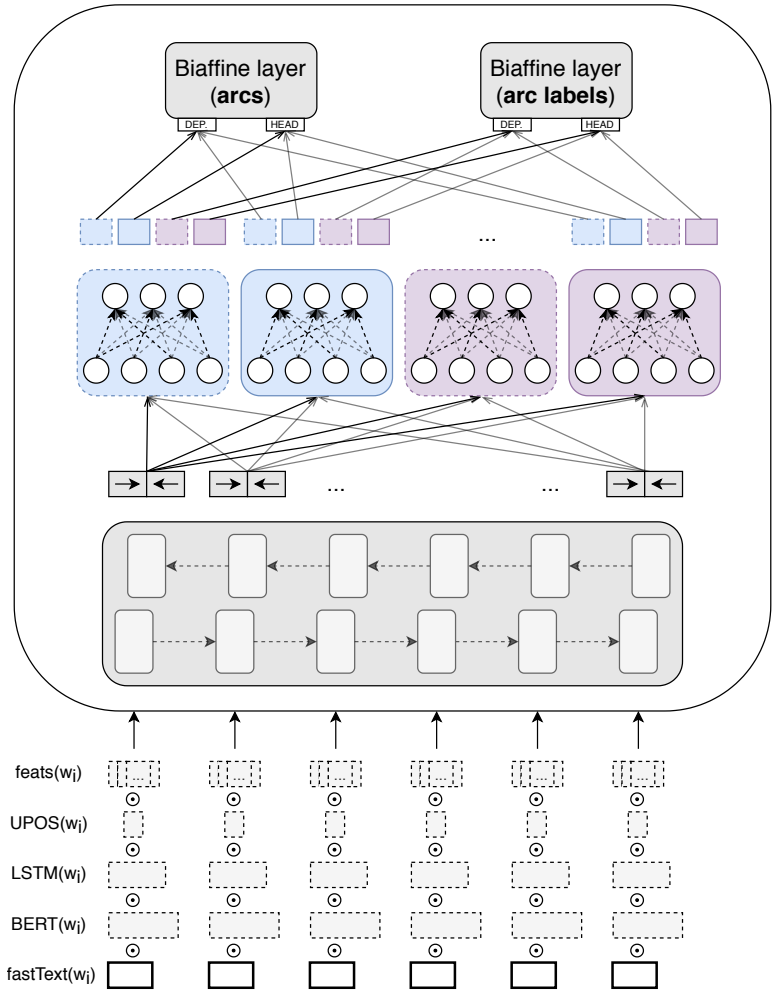


Figure 2: The deep biaffine graph-based dependency parser along with our enhancements at the input level. The dotted border of input embedding vectors, POS vectors, and morphological features (feats) is optional and varies across experiments. The $\odot$ symbol between layers represents the concatenation operation. The w_i symbol stands for token i; tokens enter the LSTM model sequentially, and we show the unrolled network.

5.1 Experimental settings

The experimental settings differ between the three evaluation tasks, so we describe them separately for each task, starting with the NER task, and followed by the DP and NER tasks.

5.1.1 Experimental settings for NER

For NER, we train each BERT model for 10 epochs and each LSTM model for 50 epochs. These parameters were determined during preliminary testing on the Slovene dataset. The selected numbers of epochs are chosen to balance the performance and training times of the models. All NER models are evaluated using 10-fold cross-validation.

We evaluate the models with the F_1 score, which is a harmonic mean of precision and recall measures. This measure is typically used in NER in a way that precision and recall are calculated separately for each of the three entity classes (location, organization, and person, but not "no entity"). For each metric, we compute the weighted average over the

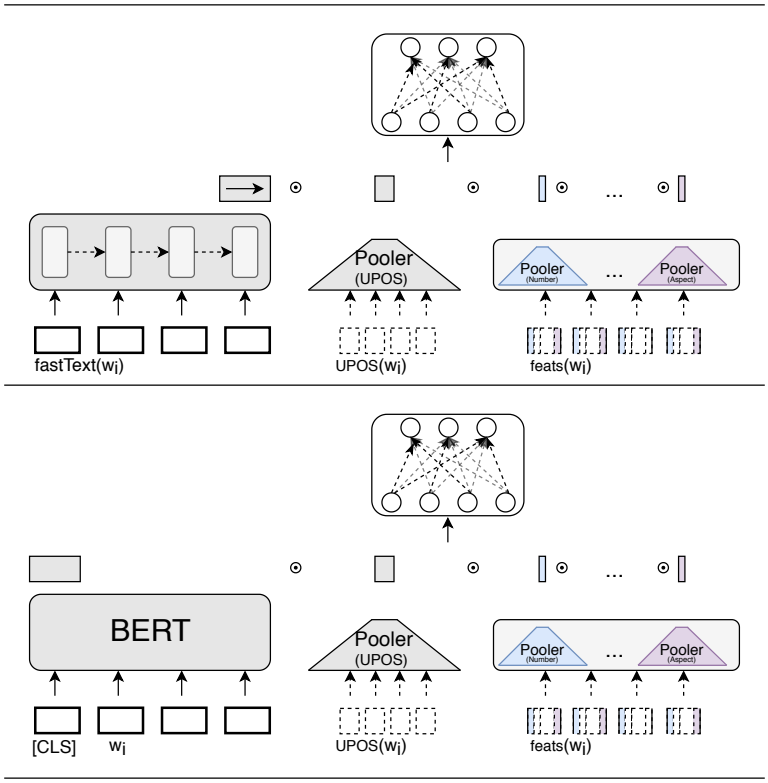


Figure 3: The baseline LSTM (top) and BERT (bottom) models for the NER task along with our modifications with morphological information. The dotted border of UPOS vectors and morphological features (feats) marks that their use is optional and varies across experiments. The $\odot$ symbol between layers represents the concatenation operation. The w_i symbol stands for token i; in case of LSTM, tokens enter the model sequentially and we show the unrolled network, while BERT processes all token simultaneously.

class scores using the frequencies of class values in the datasets as the weights. Ignoring the "no entity" (OTHR) label is a standard approach in the NER evaluation and disregards words that are not annotated with any of the named entity tags, i.e. the evaluation focuses on the named entities.

5.1.2 Experimental settings for dependency parsing

We train each DP model for a maximum of 10 epochs, using an early stopping tolerance of 5 epochs in the process. All models are evaluated using predefined splits into training, validation and testing set, determined by the respective treebank authors and maintainers (see Table 2). The final models, evaluated on the test set, are selected based on the maximum mean of unlabeled and labelled attachment scores (UAS and LAS) on the validation set. The UAS and LAS are standard accuracy metrics in DP. The UAS score is defined as the proportion of tokens that are assigned the correct syntactic head, while the LAS score is the proportion of tokens that are assigned the correct syntactic head as well as the dependency label [Jurafsky and Martin, 2009].

5.1.3 Experimental settings for comment filtering

In our NER experiments, we train BERT models for a maximum of 10 epochs and LSTM models for a maximum of 50 epochs, using early stopping with the tolerance of 5 epochs.

We evaluate the models using the classification accuracy metric. The choice reflects the fact that the distributions of class labels are approximately balanced in our datasets. We use fixed training, validation and test sets, obtained by randomly dividing the datasets in the ratio 60%:20%:20%. The initial results in the Croatian hate speech experiments

showed large variance; therefore, we repeated each experiment five times. For this task and Croatian language, we report the obtained mean classification accuracy and its standard deviation.

5.2 Experimental results

In this section, we compare the results of baseline models with their enhancements using additional morphological information. We split the presentation into three parts, according to the evaluation task: NER, DP, and CF.

5.2.1 Results for the NER task

For the NER evaluation task, we present results of the baseline NER models and models enhanced with the POS tags and morphological features, as introduced in Section 4.1. Table 3 shows the results for LSTM and BERT models. We compute the statistical significance of the differences between the baseline LSTM models and their best-performing counterpart with morphological additions. We use Wilcoxon signed-rank test [Wilcoxon et al., 1970] and underline the statistically significant differences at $p = 0.01$ level. We do not perform the significance of differences tests for BERT as the differences for those models are marginal.

As expected, the baseline models involving BERT outperform their LSTM counterparts across all languages by a large margin. When adding POS tags or universal features to LSTM-based models, we observe a noticeable increase in performance over the baselines for seven languages. For three of them (Croatian, Russian, and Swedish), the increase in F_1 score is under one per cent. For three of them (Estonian, Finnish, and Slovene), the differences in performance are statistically significant, and increases range from 1.4% (Finnish) to 4.6% (Slovene). The only language for which there is no improvement with the LSTM-based models is English.

In BERT-based models, the additional morphological features do not seem to make a practical difference. For all languages except Slovene, the increase in F_1 values over the baseline is under 0.5%. However, since preliminary tests in this task were performed on Slovene, the larger observed increase might be due to manual overfitting.

Table 3: F_1 scores for different models on the NER task in different languages. The upper part of the table shows LSTM models and the lower part shows BERT models. The best scores for each language are marked with the bold typeface separately for LSTM and BERT models. Best scores for which the difference to the baseline is statistically significant are underlined.

Model	CRO	ENG	EST	FIN	LAT	RUS	SLO	SWE
fT + LSTM	0.700	**0.890**	0.769	0.814	0.573	0.752	0.651	0.785
fT + LSTM + UPOS	**0.708**	0.889	**0.792**	0.824	**0.590**	0.754	0.692	0.787
fT + LSTM + UPOS + feats	0.706	0.883	0.781	**0.828**	0.588	0.759	**0.697**	0.785
fT + LSTM + feats	0.697	0.885	0.777	0.826	0.581	**0.760**	0.669	**0.793**
BERT	0.874	0.948	0.875	0.926	0.766	0.868	0.848	0.885
BERT + UPOS	**0.875**	0.948	0.874	0.927	**0.773**	**0.870**	**0.857**	**0.886**
BERT + UPOS + feats	**0.875**	**0.949**	**0.879**	0.927	0.763	0.869	0.853	0.883
BERT + feats	0.874	0.947	0.874	**0.928**	0.769	0.869	0.851	0.878

5.2.2 Results for the dependency parsing task

For the DP evaluation task on different languages (described in Section 4.2), we present UAS and LAS accuracies in Tables 4 and 5, respectively. We first show the scores of the baseline models (multi-layer bidirectional LSTM network with biaffine attention) followed by these models enhanced with additional inputs: LSTM or BERT contextual embeddings, POS tags and morphological features, introduced in Section 4.2. The results in the upper part of Section 4.2 represent enhancements incorporating LSTM embeddings, and in the lower part, we show the enhancements including BERT embeddings. We also statistically test the differences in UAS and LAS scores between the best performing enhanced variants and their baselines without morphological additions. As the splits are fixed in the DP tasks, we use Z-test for the equality of two proportions [Kanji, 2006] at $p = 0.01$ level. The null hypothesis is that the scores of compared models are equal. For languages where the null hypothesis can be rejected, we underline the respective best result.

Similarly to the other two evaluation tasks, the models involving BERT embeddings outperform the baselines involving LSTM embeddings on all languages by a large margin. Enhancements with LSTM embeddings improve over parsers without contextual embeddings. Models with added POS tags or universal features noticeably improve over baselines

with only LSTM embeddings in both UAS and LAS score for all languages. The increase ranges between 1.2% (Russian) and 6.8% (Lithuanian) for UAS and between 2.4% (Russian) and 10.42% (Lithuanian) for LAS. All compared differences between the LSTM baselines and the best enhanced variants are statistically significant at $p = 0.01$.

Contrary to the results observed on the other two tasks, the addition of morphological features to the baseline models with BERT embeddings improves the performance scores for all languages. For some languages, the increase seems to be practically insignificant (under one per cent). However, for six languages out of nine, the increase in UAS is over one per cent and statistically significant; the same is true for eight languages out of nine for the LAS score. For these languages, the improvement ranges from 1.1% (Finnish) to 1.8% (Lithuanian) for UAS, and from 1.1% (Croatian) to 4.4% (Lithuanian) for LAS. The only language for which both UAS and LAS scores are practically equal to the baselines with BERT embeddings is Russian.

Table 4: UAS scores for different models on the DP task in different languages. The upper part of the table shows baseline parser with enhancements using LSTM-based contextual embeddings; the lower part shows baseline parser enhanced with variants of BERT embeddings. The best scores for each language are marked with the bold typeface separately for LSTM and BERT embeddings extensions. Statistically significant differences in best scores at $p = 0.01$ level are underlined.

Model	CRO	ENG	EST	FIN	LAT	LIT	RUS	SLO	SWE
baseline	84.83	83.54	79.35	82.45	81.51	70.51	83.04	86.66	80.68
+LSTM	86.48	84.87	81.92	84.13	83.63	71.82	85.02	86.92	84.03
+LSTM+UPOS	87.81	**87.26**	85.30	86.82	86.73	75.83	85.75	91.99	86.78
+LSTM+UPOS+f.	**87.82**	87.21	**86.12**	**87.64**	**87.97**	**78.62**	86.19	**92.42**	**87.19**
+LSTM+feats	87.52	85.97	84.31	85.23	86.09	78.10	**86.26**	90.67	84.90
+mBERT	91.72	91.43	88.02	89.99	87.87	79.79	90.32	93.66	90.16
+mBERT+UPOS	**92.17**	**91.64**	89.26	90.53	89.11	80.48	90.53	94.70	91.21
+mBERT+UPOS+f.	91.97	91.56	**89.51**	**91.09**	**89.63**	**81.64**	**90.65**	**94.95**	**91.46**
+mBERT+feats	91.72	91.35	88.32	90.33	88.92	81.61	90.64	94.42	90.64

Table 5: LAS scores for different models on the DP task in different languages. The upper part of the table shows baseline parser with enhancements using LSTM-based contextual embeddings; the lower part shows baseline parser with enhancements using BERT embeddings. The best scores for each language are marked with the bold typeface separately for LSTM and BERT embedding extensions. Statistically significant differences in best scores at $p = 0.01$ level are underlined.

Model	CRO	ENG	EST	FIN	LAT	LIT	RUS	SLO	SWE
baseline	77.55	79.28	73.17	77.09	75.77	62.23	77.22	82.12	75.06
+LSTM	79.59	80.69	76.24	78.60	78.08	63.65	79.24	83.03	78.97
+LSTM+UPOS	81.95	**84.68**	81.36	83.06	82.75	70.45	80.88	88.75	82.65
+LSTM+UPOS+f.	**82.84**	84.48	**83.46**	**83.97**	**84.27**	**74.07**	**81.68**	**90.32**	**83.17**
+LSTM+feats	82.00	82.96	81.00	81.16	81.81	72.40	81.64	88.38	80.31
+mBERT	86.37	88.09	84.30	86.07	83.02	72.62	85.99	91.67	86.67
+mBERT+UPOS	**87.63**	**89.35**	86.72	87.32	85.43	74.99	86.24	92.90	87.80
+mBERT+UPOS+f.	87.44	89.13	**87.19**	**87.99**	**86.16**	**77.04**	**86.56**	**93.40**	**88.38**
+mBERT+feats	86.81	88.81	85.72	87.07	85.19	76.33	86.37	92.74	87.25

5.2.3 Results for the comment filtering task

Table 6 shows the NER results for LSTM and BERT baselines and their enhancements, described in Section 4.3. All BERT-based models outperform the LSTM-based models by a large margin for both languages, Croatian and English.

Adding POS tags and universal features to neural architectures of either type does not increase the classification accuracy and may even decrease it in some cases. As the accuracy does not increase with the best set of hyperparameters, we do not further discuss the effect of different pooling types on the performance, though we note that none of the pooling approaches consistently performed best.

Table 6: Classification accuracies for baseline and enhanced models on the NER task in different languages. The upper part of the table shows LSTM models, and the lower part shows BERT models. The best scores for each language are marked with the bold typeface separately for LSTM and BERT models. We report the mean and standard deviation over five random splits of the data. The differences between the models of the same type and language are marginal.

Model	CRO Hate speech	ENG Toxic comments
fastText + LSTM	$\mathbf{77.69 \pm 0.47}\%$	$88.14 \pm 0.12\%$
fastText + LSTM + uPOS	$76.63 \pm 0.77\%$	$88.16 \pm 0.15\%$
fastText + LSTM + uPOS + feats	$75.64 \pm 1.21\%$	$86.90 \pm 1.20\%$
fastText + LSTM + feats	$76.52 \pm 0.99\%$	$\mathbf{88.25 \pm 0.27}\%$
mBERT	$86.20 \pm 0.20\%$	$92.23 \pm 0.12\%$
mBERT + uPOS	$85.05 \pm 0.49\%$	$\mathbf{92.33 \pm 0.11}\%$
mBERT + uPOS + feats	$\mathbf{86.34 \pm 0.28}\%$	$92.29 \pm 0.20\%$
mBERT + feats	$86.22 \pm 0.35\%$	$92.20 \pm 0.13\%$

5.3 Ablation study

To further analyze different aspects of the proposed morphological enhancements, we conducted several ablation studies on the DP task where the datasets and evaluation settings allow many experiments. Similarly as in section 5.2.2, we statistically evaluate differences in performance between the baseline model and the best enhancement using the Z-test for the equality of two proportions. In cases where the null hypothesis can be rejected at $p = 0.01$ level, we underline the respective compared score. We test the impact of additional training time, quality of morphological information, and different variants of BERT models.

5.3.1 Additional training time

To test if the observed differences in performance are due to random variation in the training of models which could be reduced with longer training time, we increase the maximum training time from 10 to 15 epochs. From the studied languages, we arbitrarily choose a subset of 6 languages on which we experiment. The results are shown in Table 7.

Table 7: UAS/LAS scores achieved by models that are trained for up to 5 additional epochs (a maximum training time of 15 epochs instead of 10). The results for 10 epochs are presented in Tables 4 and 5.

Model	CRO	ENG	EST	FIN	SLO	RUS
baseline+LSTM	86.22/79.49	85.40/81.47	82.28/76.72	85.30/80.28	88.31/84.72	85.38/79.79
baseline+LSTM+UPOS	**88.43**/82.72	87.30/84.74	85.35/81.52	87.27/83.77	92.11/89.19	86.46/81.35
baseline+LSTM+UPOS+f.	88.26/**83.26**	**87.92**/**85.26**	**86.40**/**83.65**	**87.83**/**84.22**	**92.64**/**90.64**	86.68/82.18
baseline+LSTM+feats	88.08/82.74	86.61/83.58	84.50/81.24	85.71/81.95	90.73/88.60	**86.75**/**82.21**
baseline+mBERT	92.06/87.03	91.10/87.77	88.06/84.60	90.58/87.06	93.97/92.11	90.26/86.15
baseline+mBERT+UPOS	92.03/87.55	**91.77**/**89.56**	89.35/86.60	91.16/87.98	94.54/92.80	91.05/87.04
baseline+mBERT+UPOS+f.	**92.09**/**87.71**	91.74/89.49	**89.49**/**87.26**	**91.32**/**88.42**	**95.16**/**93.78**	**91.11**/**87.23**
baseline+mBERT+feats	91.93/86.97	91.33/88.89	88.37/85.78	90.85/87.85	94.49/93.08	90.79/86.64

We can observe that longer training times slightly increases the scores for all model variants, though their relative order stays the same. The models with added morphological features still achieve better results, so the performance increases do not seem to be the effect of random fluctuations in training due to the different amount of training steps.

5.3.2 Quality of morphological information

In the second ablation study, we evaluate the impact of the quality of morphological information. We replace the high quality (human-annotated) POS tags and morphological features used in Section 4.2 with those predicted by machine learning models. In this way, we test a realistic setting where the morphological information is at least to a certain degree noisy. We obtain POS tags and morphological features from Stanza models prepared for the involved languages [Qi et al., 2020]. To avoid overly optimistic results, we make sure to use models that are not trained on the same datasets used in our DP experiments. This is possible for a subset of five languages. Table 8 shows the results of DP models, trained with predicted morphological features. The first two lines of the table show the quality of used features,

expressed as the proportion of tokens, for which POS tags and *all* morphological features are correctly predicted, i.e. we report their accuracy.

The general trend is that using predicted features results in much smaller (best case) performance increases, though some languages still see significant increases. For LSTM models, the increases range from 0.42% (English) to 1.60% (Slovene) for UAS, and from 0.67% (English) to 2.41% (Finnish) for LAS. For BERT models, the increases are marginal and range from 0.00% (English) to 0.49% (Finnish) for UAS, and from -0.10% (i.e. decrease, Slovene) to 0.66% (Finnish) for LAS. These results are consistent with the results of our NER experiments in section 5.2.1, where we have no access to human-annotated features and find that noisy features only help LSTM-based models.

As noisy features might require different training, we also tried to increase the maximal number of training steps to 15 epochs and repeated the tuning of hyperparameters for one language (Estonian). None of these changes improved the scores by a practically relevant margin.

These results indicate that adding predicted morphological features to models with BERT embeddings might not be practically useful since their quality needs to be very high. However, since human annotated morphological features improve the performance on the DP task, this suggests that there is a room for improvement in BERT pre-training. It seems that pre-training tasks of BERT (masked language modelling and next sentence prediction) do not fully capture the morphological information present in the language.

Table 8: UAS/LAS scores achieved by models that are trained with predicted (noisy) instead of human-annotated morphological features. In the first two lines, we report the proportion of tokens for which UPOS tags and UPOS features are correctly predicted.

	ENG	**EST**	**FIN**	**SLO**	**RUS**
UPOS accuracy (%):	92.45	91.15	87.59	80.45	89.32
feats accuracy (%):	93.92	88.86	86.20	78.74	85.22
baseline+LSTM	84.87/80.69	81.92/76.24	84.13/78.60	86.92/83.03	85.02/79.24
baseline+LSTM+UPOS	85.18/**81.36**	**82.44**/77.15	85.33/80.61	88.14/84.06	84.90/79.53
baseline+LSTM+UPOS+f.	85.04/81.02	82.24/**77.39**	**85.56**/**81.01**	**88.52**/**85.00**	**85.82**/**80.89**
baseline+LSTM+feats	**85.29**/81.22	81.42/76.19	84.89/80.43	88.12/84.12	84.74/79.56
baseline+mBERT	**91.43**/88.09	88.02/84.30	89.99/86.07	93.66/**91.67**	90.32/85.99
baseline+mBERT+UPOS	**91.43**/**88.33**	**88.20**/**84.69**	90.05/86.33	93.38/91.42	**90.59**/**86.17**
baseline+mBERT+UPOS+f.	91.08/88.05	88.06/84.39	**90.48**/**86.73**	**93.73**/91.57	90.34/85.74
baseline+mBERT+feats	90.90/87.74	87.88/84.42	89.77/85.91	93.37/91.23	89.65/85.27

5.3.3 Specific BERT models

In the third ablation study, we replace the embeddings obtained from the multilingual uncased BERT model with those obtained from more specific multilingual BERT models and monolingual BERT models. In experiments involving multilingual BERT models, we use Croatian/Slovene/English CroSloEngual BERT and Finnish/Estonian/English FinEst BERT [Ulčar and Šikonja, 2020]. In experiments with monolingual BERT models, we use English bert-base-uncased [Devlin et al., 2019], Finnish bert-base-finnish-cased-v1 [Virtanen et al., 2019], and Russian RuBert [Kuratov and Arkhipov, 2019]. We only perform the experiments for a subset of studied languages for which we were able to find more specific BERT models. The aim is to check if the additional morphological features improve the performance of more language-specific BERT models. These are trained on a lower number of languages, and larger amounts of texts in the involved languages compared to the original multilingual BERT model Devlin et al. [2019]. Due to this language-specific training, we expect these BERT models to capture the nuances of the languages better, thus possibly benefiting less from the additional morphological features. The results are shown in Table 9 for trilingual BERT models and in Table 10 for monolingual BERT models.

In most cases, the specific multilingual BERT models without additional features do as well as or better than the best performing original multilingual BERT model with additional features. The only worse LAS scores are achieved on English and Estonian, indicating that trilingual BERT models are better adapted to the task than the original multilingual models. The addition of morphological features increases the UAS and LAS even further. For UAS, the improvements are generally smaller than before (see Figure 2), with the only improvement larger than 1% being on Estonian. For LAS, the improvements are larger compared to UAS and range from 0.54% (Slovene) to 2.23% (Estonian); only one language sees an improvement of less than 1%. These results indicate that the additional morphological features still contain valuable information for the DP task, which the BERT models do not capture. We hypothesise that slightly

lower performance of FinEst BERT (without additions) on Estonian as well as slightly larger increase when adding morphological features might be due to variability in the training process.

For monolingual models (the most language-specific), the results are mixed, but the general trend is that the additional features help little or not at all. The English monolingual model without and with morphological additions performs comparably to its multilingual (CroSloEngual) counterpart, and we do not see any additional increase in the performance. The Russian monolingual model achieves comparable UAS and LAS to the best performing original multilingual BERT model with additional features. With the addition of morphological features, the scores increase, but only marginally (under 0.5%). The Finnish monolingual model outperforms the best performing multilingual (FinEst) counterpart. The addition of morphological features to the monolingual Finnish model increases the scores further: UAS by 0.33% and LAS by 0.68%, but these increases are not statistically significant.

Table 9: UAS/LAS scores achieved by trilingual BERT models, each trained on a smaller set of languages (3) than the original multilingual BERT model (104).

BERT variant	CroSloEngual BERT			FinEst BERT	
Model/Language	CRO	ENG	SLO	FIN	EST
baseline+BERT	92.63/87.84	91.69/88.37	95.48/93.98	92.61/89.35	89.84/86.51
baseline+BERT+UPOS	**93.29**/**89.01**	**92.05**/**89.69**	95.72/94.27	**93.39**/**90.66**	**91.06**/88.67
baseline+BERT+UPOS+f.	92.95/88.33	91.99/89.65	**95.82**/**94.52**	93.23/90.61	91.01/**88.74**
baseline+BERT+feats	92.87/87.98	91.27/88.58	95.33/94.05	93.12/90.54	89.94/87.44

Table 10: UAS/LAS achieved by monolingual BERT models.

BERT variant	bert-base-uncased	bert-base-finnish-cased-v1	RuBert
Model	ENG	FIN	RUS
baseline + BERT	91.82/88.74	94.20/91.51	90.83/86.39
baseline + BERT + UPOS	**91.92**/89.60	**94.53**/**92.19**	**91.24**/**87.31**
baseline + BERT + UPOS + f.	91.81/**89.62**	94.08/91.73	91.04/87.04
baseline + BERT + feats	91.23/88.76	94.09/91.80	90.88/86.61

6 Conclusion

We analysed adding explicit morphological information in the form of embeddings for POS tags and morphological features to two currently dominant neural network architectures used in NLP: LSTM networks and transformer-based BERT models. We compared models enhanced with morphological information with baselines on three tasks (NER, DP, and NER). To obtain general conclusions, we used subsets of eight morphologically-rich languages from different language families.

The results indicate that adding morphological information to NER prediction models is not beneficial, but it improves the performance in the NER and DP tasks. For the DP task, the improvement depends on the quality of the morphological features. The additional morphological features consistently benefited LSTM-based models for NER and DP, both when they were of high quality and predicted (noisy). For BERT-based models, the *predicted* features do not make any practical difference for the NER and DP task but improve the performance in the DP task when they are of *high quality*. Testing different variants of BERT shows that language specialised variants improve the performance on the DP task and the additional morphological information is beneficial, though less and less as we shift from multilingual towards monolingual models.

The comparison of different BERT variants indicates that BERT models do not completely capture the language morphology. Since the release of BERT, several new pre-training objectives have been proposed, such as syntactic and semantic phrase masking [Zhou et al., 2020b] and span masking [Joshi et al., 2020]. In further work, it makes sense to apply these models to the DP task in order to test how well they capture the morphology. Further, the effect of morphological features could be analysed on additional tasks and languages, since the explicit morphological information does not seem to benefit them equally.

Acknowledgements

This paper is supported by European Union's Horizon 2020 Programme project EMBEDDIA (Cross-Lingual Embeddings for Less-Represented Languages in European News Media, grant no. 825153). The research was supported by the Slovene Research Agency through research core funding no. P6-0411. The Titan X Pascal used for a part of this research was donated by the NVIDIA Corporation.

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

Danqi Chen and Christopher Manning. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750, Doha, Qatar, October 2014. Association for Computational Linguistics.

Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, page 160–167, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781605582054.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2126–2136, Melbourne, Australia, July 2018. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

Timothy Dozat and Christopher D. Manning. Deep biaffine attention for neural dependency parsing, 2016.

Daniel Edmiston. A systematic analysis of morphological content in BERT models for multiple languages. arXiv:2004.03032, 2020.

Yanai Elazar, Shauli Ravfogel, Alon Jacovi, and Yoav Goldberg. When Bert forgets how to POS: Amnesic probing of linguistic properties and MLM predictions, 2020.

Paula Fortuna and Sérgio Nunes. A survey on automatic detection of hate speech in text. *ACM Comput. Surv.*, 51(4), July 2018. ISSN 0360-0300.

Lei Gao and Ruihong Huang. Detecting online hate speech using context aware models. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017*, pages 260–266, Varna, Bulgaria, September 2017. INCOMA Ltd.

Spiros V Georgakopoulos, Sotiris K Tasoulis, Aristidis G Vrahatis, and Vassilis P Plagianakos. Convolutional neural networks for toxic comment classification. In *Proceedings of the 10th Hellenic Conference on Artificial Intelligence*, pages 1–6, 2018.

Onur Güngör, Eray Yıldız, Suzan Uskudarli, and Tunga Gungor. Morphological embeddings for named entity recognition in morphologically rich languages. *arXiv preprint arXiv:1706.00506*, 2017.

Jiahui Han, Shengtan Wu, and Xinyu Liu. jhan014 at SemEval-2019 task 6: Identifying and categorizing offensive language in social media. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 652–656, Minneapolis, Minnesota, USA, June 2019. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, November 1997. ISSN 0899-7667.

Zhiheng Huang, Wei Xu, and Kai Yu. Bidirectional LSTM-CRF models for sequence tagging. *ArXiv*, abs/1508.01991, 2015.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. What does BERT learn about the structure of language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3657, Florence, Italy, July 2019. Association for Computational Linguistics.

Tao Ji, Yuanbin Wu, and Man Lan. Graph-based dependency parsing with graph neural networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2475–2485, Florence, Italy, July 2019. Association for Computational Linguistics.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. SpanBERT: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8: 64–77, 2020.

Daniel Jurafsky and James H. Martin. *Speech and Language Processing (2nd Edition)*. Prentice-Hall, Inc., USA, 2009. ISBN 0131873210.

Gopal K Kanji. *100 statistical tests*. Sage, 2006.

Jurgita Kapočiūtė-Dzikienė, Joakim Nivre, and Algis Krupavičius. Lithuanian dependency parsing with rich morphological features. In *Proceedings of the fourth workshop on statistical parsing of morphologically-rich languages*, pages 12–21, 2013.

Mojtaba Khallash, Ali Hadian, and Behrouz Minaei-Bidgoli. An empirical study on the effect of morphological and lexical features in Persian dependency parsing. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 97–107, Seattle, Washington, USA, October 2013. Association for Computational Linguistics.

Eliyahu Kiperwasser and Yoav Goldberg. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association for Computational Linguistics*, 4:313–327, 2016.

Dan Kondratyuk and Milan Straka. 75 languages, 1 model: Parsing universal dependencies universally. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2779–2795, 2019.

Artur Kulmizev, Miryam de Lhoneux, Johannes Gontrum, Elena Fano, and Joakim Nivre. Deep contextualized word embeddings in transition-based and graph-based dependency parsing - a tale of two parsers revisited. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2755–2768, 2019.

Yuri Kuratov and Mikhail Arkhipov. Adaptation of deep bidirectional multilingual transformers for Russian language, 2019.

Onur Kuru, Ozan Arkan Can, and Deniz Yuret. CharNER: Character-level named entity recognition. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 911–921, Osaka, Japan, December 2016. The COLING 2016 Organizing Committee.

Zuchao Li, Jiaxun Cai, Shexia He, and Hai Zhao. Seq2seq dependency parsing. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3203–3214, Santa Fe, New Mexico, USA, August 2018. Association for Computational Linguistics.

Yongjie Lin, Yi Chern Tan, and Robert Frank. Open Sesame: Getting inside BERT's linguistic knowledge, 2019.

Shervin Malmasi and Marcos Zampieri. Detecting hate speech in social media. In *Proceedings of the International Conference Recent Advances in Natural Language Processing*, pages 467–472, 2017.

Yuval Marton, Nizar Habash, and Owen Rambow. Improving Arabic dependency parsing with lexical and inflectional morphological features. In *Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 13–21, Los Angeles, CA, USA, June 2010. Association for Computational Linguistics.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 523–530, Vancouver, British Columbia, Canada, October 2005. Association for Computational Linguistics.

Kristian Miok, Dong Nguyen-Doan, Blaž Škrlj, Daniela Zaharie, and Marko Robnik-Šikonja. Prediction uncertainty estimation for hate speech classification. In *International Conference on Statistical Language and Speech Processing*, pages 286–298. Springer, 2019.

Joakim Nivre. An efficient algorithm for projective dependency parsing. In *Proceedings of the Eighth International Conference on Parsing Technologies*, pages 149–160, Nancy, France, April 2003.

Joakim Nivre, Mitchell Abrams, Željko Agić, et al. Universal Dependencies 2.6, 2020. URL `http://hdl.handle.net/11234/1-2988`. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Wenzhe Pei, Tao Ge, and Baobao Chang. An effective neural network model for graph-based dependency parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 313–322, Beijing, China, July 2015. Association for Computational Linguistics.

Telmo Pires, Eva Schlinger, and Dan Garrette. How multilingual is multilingual BERT? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4996–5001, Florence, Italy, July 2019. Association for Computational Linguistics.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. Stanza: A Python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 2020.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. A primer in BERTology: What we know about how BERT works, 2020.

Tatjana Scheffler, Erik Haegert, Santichai Pornavalai, and Mino Lee Sasse. Feature explorations for hate speech classification. In *14th Conference on Natural Language Processing KONVENS 2018*, volume 6, page 8, 2018.

Wolfgang Seeker and Jonas Kuhn. On the role of explicit morphological feature representation in syntactic dependency parsing for German. In *Proceedings of the 12th International Conference on Parsing Technologies*, pages 58–62, Dublin, Ireland, October 2011. Association for Computational Linguistics.

Serhiy Shtovba, Olena Shtovba, and Mykola Petrychko. Detection of social network toxic comments with usage of syntactic dependencies in the sentences. In *Proceedings of the Second International Workshop on Computer Modeling and Intelligent Systems*, pages 313–323, 2019.

Lilia Simeonova, Kiril Simov, Petya Osenova, and Preslav Nakov. A morpho-syntactically informed LSTM-CRF model for named entity recognition, 2019.

Milan Straka and Jana Straková. Tokenizing, POS tagging, lemmatizing and parsing UD 2.0 with UDPipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 88–99, Vancouver, Canada, August 2017. Association for Computational Linguistics.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy, July 2019. Association for Computational Linguistics.

Matej Ulčar and Marko Robnik Šikonja. FinEst BERT and CroSloEngual BERT: less is more in multilingual models, 2020.

Cynthia Van Hee, Els Lefever, Ben Verhoeven, Julie Mennes, Bart Desmet, Guy De Pauw, Walter Daelemans, and Véronique Hoste. Detection and Fine-Grained Classification of Cyberbullying Events. In *Proceedings of the International Conference Recent Advances in Natural Language Processing*, pages 672–680, 2015.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, page 6000–6010, 2017.

Antti Virtanen, Jenna Kanerva, Rami Ilo, Jouni Luoma, Juhani Luotolahti, Tapio Salakoski, Filip Ginter, and Sampo Pyysalo. Multilingual is not enough: BERT for Finnish, 2019.

Frank Wilcoxon, SK Katti, and Roberta A Wilcox. Critical values and probability levels for the wilcoxon rank sum test and the wilcoxon signed rank test. *Selected tables in mathematical statistics*, 1:171–259, 1970.

Hiroyasu Yamada and Yuji Matsumoto. Statistical dependency analysis with support vector machines. In *Proceedings of the Eighth International Conference on Parsing Technologies*, pages 195–206, Nancy, France, April 2003.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. SemEval-2019 task 6: Identifying and categorizing offensive language in social media (OffensEval). In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 75–86, Minneapolis, Minnesota, USA, June 2019. Association for Computational Linguistics.

Marcos Zampieri, Preslav Nakov, Sara Rosenthal, Pepa Atanasova, Georgi Karadzhov, Hamdy Mubarak, Leon Derczynski, Zeses Pitenis, and Çağrı Çöltekin. SemEval-2020 task 12: Multilingual offensive language identification in social media (OffensEval 2020), 2020.

Houquan Zhou, Yu Zhang, Zhenghua Li, and Min Zhang. Is POS tagging necessary or even helpful for neural dependency parsing?, 2020a.

Junru Zhou, Zhuosheng Zhang, Hai Zhao, and Shuailiang Zhang. LIMIT-BERT: Linguistic informed multi-task BERT, 2020b.

Appendix B: To BAN or Not to BAN: Bayesian Attention Networks for Reliable Hate Speech Detection

Noname manuscript No.
(will be inserted by the editor)

To BAN or Not to BAN: Bayesian Attention Networks for Reliable Hate Speech Detection

Kristian Miok · Blaž Škrlj · Daniela Zaharie · Marko Robnik-Šikonja

the date of receipt and acceptance should be inserted later

Abstract Background Hate speech is an important problem in the management of user-generated content. To remove offensive content or ban misbehaving users, content moderators need reliable hate speech detectors. Recently, deep neural networks based on the transformer architecture, such as the (multilingual) BERT model, achieve superior performance in many natural language classification tasks, including hate speech detection. So far, these methods have not been able to quantify their output in terms of reliability.

Methods We propose a Bayesian method using Monte Carlo dropout within the attention layers of the transformer models to provide well-calibrated reliability estimates. We evaluate and visualize the results of the proposed approach on hate speech detection problems in several languages. Additionally, we test if affective dimensions can enhance the information extracted by the BERT model in hate speech classification.

Corresponding author

Kristian Miok (✉)
West University of Timisoara, Computer Science Department
Bulevardul Vasile Pârvan 4, 300223 Timisoara, Romania
E-mail: kristian.miok@e-uvt.ro

Blaž Škrlj
Jožef Stefan International Postgraduate School, and Jožef Stefan Institute,
Jamova 39, 1000 Ljubljana, Slovenia
E-mail: blaz.skrlj@ijs.si

Daniela Zaharie
West University of Timisoara, Computer Science Department
Bulevardul Vasile Pârvan 4, 300223 Timisoara, Romania
E-mail: daniela.zaharie@e-uvt.ro

Marko Robnik-Šikonja
University of Ljubljana, Faculty of Computer and Information Science,
Večna pot 113, 1000 Ljubljana, Slovenia
E-mail: marko.robnik@fri.uni-lj.si

Results Our experiments show that Monte Carlo dropout provides a viable mechanism for reliability estimation in transformer networks. Used within the BERT model, it offers state-of-the-art classification performance and can detect less trusted predictions. Also, it was observed that affective dimensions extracted using sentic computing methods can provide insights toward interpretation of emotions involved in hate speech.

Conclusions Our approach not only improves the classification performance of the state-of-the-art multilingual BERT model but the computed reliability scores also significantly reduce the workload in an inspection of offending cases and reannotation campaigns. The provided visualization helps to understand the borderline outcomes.

Keywords prediction uncertainty · reliability estimation · Monte Carlo dropout · transformer neural networks · Bayesian BERT · Sentic Computing · model calibration

1 Introduction

With the rise of social network popularity, hate speech phenomena have significantly increased [14]. Hate speech not only harms both minority groups and the whole society, but it can lead to actual crimes [3]. Thus, (automated) hate speech detection mechanisms are urgently needed. However, falsely accusing people of hate speech is also a problem. Many content providers rely on human moderators to reliably decide if a given text is offensive or not, but this is a mundane and stressful job which can even cause post-traumatic stress disorders[1]. There have been many attempts to automate the detection of hate speech in social media using machine learning, but existing models lack the quantification of reliability for their decisions.

In the last few years, recurrent neural networks (RNNs) were the most popular text classification choice. Long Short Term Memory (LSTM) networks, the most successful RNN architecture, were already successfully adapted for the assessment of predictive reliability in hate speech classification [42]. Recently, neural network architecture with attention layers, called 'transformer architecture' [59], showed even better performance on almost all language processing tasks. Using transformer networks for masked language modeling produced breakthrough pretrained models, such as BERT (Bidirectional Encoder Representations from Transformers) [15]. The attention mechanism, which is a crucial part of transformer networks, became an essential part of natural language understanding with a significant impact on language applications. We aim to investigate the behavior of the attention mechanism concerning the reliability of predictions. We focus on the hate speech recognition task.

[1] `https://www.bbc.com/news/technology-51245616`

In hate speech detection, reliable predictions are needed to remove harmful content and possibly ban malicious users without harming the freedom of speech [42]. Standard neural networks are inadequate for the assessment of predictive uncertainty, and the best solution is to use the Bayesian inference framework. However, classical Bayesian inference techniques do not scale well in neural networks with high dimensional parameter space [24]. Various methods were proposed in order to overcome this problem [45]. One of the most efficient methods is called Monte Carlo Dropout (MCD) [20]. Its idea is to use dropout in neural networks as a regularization technique [55] and interpret it as a Bayesian optimization approach that takes samples from the approximate posterior distribution.

Several authors have shown that emotional information [6] extracted from a text can improve the performance of lexical approaches and standard machine learning algorithms [35, 1, 52, 2]. The role and utility of emotional information in deep learning have not yet been established; besides, we still have only limited understanding of the emotions in the text. A series of computational models that bridge the gap between the human emotional perspective evolved in a domain known as 'Sentic Computing' [7]. The computational initiative, named 'SenticNet', combines knowledge from psycholinguists, neuroscientists, and computer scientists to better understand emotions in text. We used information on affective dimensions provided by SenticNet, together with the outputs of the state-of-the-art contextual language model BERT [15]. This was enhanced with a reliability estimation mechanism based on MCD as input for a hate speech classifier. Concerning emotions, we follow two goals in this work: i) to test the predictive performance of emotion-enhanced BERT models in hate speech detection, and ii) to better understand the role of emotions in hate speech.

Our main contributions are:

1. We present a novel methodology for the assessment of prediction uncertainty in attention networks and in BERT models.
2. Empirical analysis of the proposed Bayesian Attention Networks (BANs) and MCD enhanced BERT models show an improved calibration and prediction performance on hate speech detection tasks in several languages.
3. We combine contextual and reliability information obtained from MCD BERT with sentiment-related knowledge provided by SenticNet.
4. We demonstrate novel visualization of prediction uncertainty for individual instances, as well as for groups of instances.

The paper consists of six more sections. In Section 2, we present related works on prediction uncertainty, hate speech detection and its relationship with sentiment analysis. In Section 3, we propose the methodology for uncertainty assessment in transformer networks using attention layers and MCD, while in Section 4, we analyze the calibration of predictions. Section 5 presents the datasets and the evaluation scenario. The obtained results are presented in Section 6, followed by conclusions and ideas for further work in Section 7.

2 Related Work

We present the related work categorized into four areas. In Section 2.1, we introduce work done on hate speech detection, followed by the related research on transformer architecture for text classification in Section 2.2. In Section 2.3, we describe existing approaches for the assessment of uncertainty in text classification. Finally, in Section 2.4, we relate hate speech detection with the particularities of sentic computing.

2.1 Hate Speech Detection

Analyzing sentiments and extracting emotions from texts are very useful natural language processing (NLP) applications. With the rise of social media popularity, the hate speech detection became highly needed. Hate speech is defined as written or oral communication that abuses or threatens a specific group or target [62].

Detecting abusive language for less-resourced languages is complex, and has inspired research in multilingual and cross-lingual methods [56]. These methods are especially useful when the involved languages are morphologically or geographically close [47]. In our work, we investigate hate speech detection methods for English, Croatian, and Slovene languages. The English language is well-resourced and researched [33, 14, 63]. Recently, hate speech detection studies appeared for Croatian [27, 34, 29] and Slovene [17, 30, 60].

The hate speech detection is mostly treated as a binary text classification problem. In the past, the most frequently used classifier was the Support Vector Machines (SVM) method [54]. However, deep neural networks are now a dominant technique, first through RNNs [38], and recently using the pre-trained transformer networks [44, 64]. In this work, we analyzed the state-of-the-art pre-trained transformer networks, called (multilingual) BERT model.

2.2 Attention Networks for Text Classification

Attention mechanism is a key component of transformer architecture, proposed by Vaswani et al. [59]. Due to its power and suitability for parallelization, this architecture soon replaced LSTM networks for many NLP tasks. Recently, large pre-trained transformer models have been investigated in the context of text classification tasks. For example, Kant et al. [25] trained both multiplicative LSTM (mLSTM) and transformer language models on a large 40GB text dataset [36] and transferred those models to binary and multi-class text classification problems. They concluded that the transformer model outperforms the mLSTM model, especially when fine-tuned for multidimensional emotions classification.

The BERT model [15] uses the transformer architecture and large text corpora to learn masked language model and sequence of sentences tasks. BERT and its follow-ups are able learn and extract many language characteristics (both

syntactic and semantic) and excel for many text classification tasks. Despite the short time since its conception, BERT has already attracted enormous attention from the NLP community. Hundreds of research groups extensively research it; see a recent overview by Rogers et al. [53]. Practical guidelines on how to fine-tune the BERT model for text classification were compiled by Sun et al. [57].

A multilingual hierarchical attention mechanism for document classification was investigated by several authors [48, 16, 66]. However, different attention layers of large pre-trained models were not tested separately or in the context of prediction reliability. Also, to the best of our knowledge, the predictive reliability of BERT outputs has not been investigated, yet.

2.3 Prediction Uncertainty for Text Classification

While recent works on classification reliability mostly investigate deep neural networks, many other probabilistic classifiers were analyzed in the past [46]. For example, Platt [50] explores the probabilistic properties of SVM predictions.

Prediction uncertainty is an important issue for black-box models like neural networks, as they do not provide interpretability or reliability information about their predictions. Most reliability scores for deep neural networks are based on a Bayesian framework. The most popular exception is the work of Lakshminarayanan et al. [28], who proposed using deep ensembles to estimate the prediction uncertainty.

An efficient approach to reliability assessment in neural networks is to mimic the Bayesian inference using MCD [20]. The dropout technique was first introduced to RNNs in 2013 [61], but further research revealed a negative impact of dropout in RNNs [4]. Later, dropout was successfully applied to language modeling by Zaremba et al. [68], who applied it only to fully connected layers. Gal and Ghahramani [21] implemented the variational inference based dropout, which can regularize also recurrent layers. Additionally, they provide a solution for dropout within word embeddings. The method mimics Bayesian inference by combining probabilistic parameter interpretation and deep RNNs. The authors introduce the idea of augmenting probabilistic RNN models with the prediction uncertainty estimation. Several other works investigate how to estimate prediction uncertainty using RNNs [69], e.g., Bayes by Backpropagation (BBB) [18].

Recently, a fast and scalable method called 'SWAG' was proposed by Maddox et al. [32]. The main idea of this method is to randomize the learning rate and interpret it as a sampling from the Gaussian distribution. SWAG fits the Gaussian distribution by capturing the Stochastic Weight Averaging (SWA) mean and co-variance matrix, representing the first two moments of stochastic gradient descent iterations. Different to SWAG, we use the Gaussian distribution as a posterior over neural network weights, and then perform a Bayesian model averaging for uncertainty estimation and calibration.

MCD was recently used within several models and different architectures to obtain the prediction uncertainty and improve the classification results [40, 43, 41]. Transformer networks were not yet analyzed.

2.4 Sentic Computing

Sentiments and emotions play an essential role in hate and offensive speech, and have been used successfully in their automatic detection. Martins et al. [35] have used eight basic emotions from Plutchik's model [51], the positive and negative sentiment polarities, indicator of a presence of a word in the Hatebase lexicon[2], and the intensity of anger emotion. Their combination of the lexicon-based and machine learning approach successfully predicted hate speech and showed a high utility of emotional features. Alorainy et al., 2018 [1] used the emotional analysis on Twitter suspended accounts and discovered that they contain more disgust, negative sentiment, fear, and sadness than active accounts. Using this information for hate speech detection, their machine learning models showed improved performance. Rodríguez et al. [52] also used the eight basic emotions in their emotional analysis and showed that emotions could improve Facebook posts' clustering. Finally, Bauwelinck and Lefever [2] used several different groups of features (linguistic, sentiment, and Twitter-specific features such as hashtags and profanity lexicon) to predict hate speech. Interestingly, their results show that Twitter-specific features are the most successful, and the additional sentiment features do not improve predictive performance. All the methods mentioned above use either classical machine learning approaches such as SVM, Naive Bayes, logistic regression, and random forest, or RNNs, such as LSTMs.

To advance approaches based on lexical keywords and frequency statistics, Cambria and Hussain [7] proposed a framework for emotional computing called 'SenticNet' that captures semantics and latent emotional information by relying on the implicit meaning associated with commonsense concepts. The original emotion categorization model called Hourglass of Emotions [9] was supported by the SenticNet 4 framework [7], while its newer revised version [58] is used in SenticNet 6 framework [11]. These models are biologically-inspired and psychologically-motivated. Each of the two models is based on four independent but concomitant affective dimensions, which can be combined to build more complex emotions. Based on this, SenticNet framework can describe and explain emotional experiences by disassembling text to the ground sentiments.

The SenticNet framework has been successfully used in sentiment classification problems. Sentic LSTM [31] integrates the explicit emotional information with the LSTM networks by adding a recurrent additive network that simulates sentic patterns. A recent SenticNet 6 framework [11] combines top-down and bottom-up knowledge representation. From top-up direction it encodes meaning

[2] `http://www.hatebase.org`

using symbolic models (logic and semantic networks); in bottom-up direction it learn syntactic patterns from data, using subsymbolic methods (biLSTM and BERT). Authors report state-of-the-art results for sentiment analyses.

In our work, we use the transformer architecture that can extract highly relevant information from texts. Concerning emotions, the question we investigate is whether adding emotional information to the distribution of predictions can improve the performance of hate speech detection. This question is particularly relevant for the current state-of the-art BERT model [15], which is known to capture a plethora of language information, such as part-of-speech tags, dependency structure, and sentiment.

3 Bayesian Attention Networks

The BERT model [15] is the transformer network that has achieved state-of-the-art results in many NLP tasks, including text classification [65, 23, 13]. In this work, we introduce Monte Carlo Dropout to transformer networks and BERT to construct their Bayesian variants. Analysis of different amounts of dropout, different variants of BERT modifications, and their hyper-parameters would require pretraining and fine-tuning several different BERT models, which would require substantial computational resources. For example, pretraining a single BERT model on four TPUs requires more than a month of computational time. Thus, in this work, we explore two reliability extensions, i) the reliability on the encoder part of the BERT architecture trained from scratch (without pretraining) on the task of interest (in this work, we refer to these models as the attention networks), and ii) reliability of pre-trained BERT models, using only fine-tuning. We believe this is a reasonable setting which sheds light on an important reliability aspect of transformer networks.

The BERT model [15] is the transformer network that has achieved state-of-the-art results in many NLP tasks, including text classification [65, 23, 13]. In this work, we introduce MCD to transformer networks and BERT to construct their Bayesian variants. Analysis of different amounts of dropout, different variants of BERT modifications, and their hyper-parameters would require training several different BERT models, which would require substantial computational resources. For example, training a single BERT model on four TPUs requires more than a month of computational time. Thus, in this work, we explore the reliability on only the encoder part of the BERT architecture, called attention network, and entire pre-trained BERT models. We believe this is a reasonable setting which sheds light on an important reliability aspect of transformer networks.

In Section 3.1, we first formally define the attention network architecture, and in Section 3.2, we make it Bayesian by introducing MCD. Finally, in Section 3.3, we describe how the MCD principle can be employed in already pre-trained BERT models.

3.1 Attention Networks

The basic architecture of the attention network follows the architecture of transformer networks [59] and is shown in Figure 1. The proposed architecture

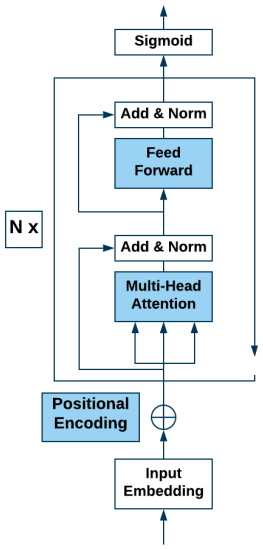


Fig. 1: A scheme of Attention Networks. The dropout is introduced in the blue colored layers.

is similar to the encoder part of the transformer architecture. The difference is in the output part, where a single output head was added to perform binary classification using the sigmoid activation function. The main difference to BERT, which also uses just the encoder part of transformer network, is that we do not use any pretraining. The second difference is that attention network uses the classification head and BERT has the language model head. In both cases the output is composed of feed-forward layers followed by the non-linearity but with different dimensions in each case. By not relying on the pretraining, we are much more flexible concerning the number of layers and number of neurons in each layer. For our tasks, we use orders of magnitude fewer parameters, e.g., we used a maximum of 3 million parameters (at the expense of loosing information from pretraining). The architecture can contain many attention heads, where a single attention head is computed as:

$$o_h = \text{softmax}(\frac{\boldsymbol{Q} \cdot \boldsymbol{K}^T}{\sqrt{d_k}}) \cdot \boldsymbol{V},$$

The attention matrices are commonly known as the query $\boldsymbol{Q}$, the key $\boldsymbol{K}$, and the value matrix $\boldsymbol{V}$. The normalizing factor, d_k, denotes the dimensionality of keys. The attention function can be described as mapping the query and the set of key-value pairs to the output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values. The weight assigned to each value is computed by a compatibility function of the query with the corresponding key.

Intuitively, the multiplication of query and key vectors with subsequent values can be understood as the extraction of *relations*. The softmax activation enables each pair of considered input tokens to be represented with a single real value. It effectively introduces *sparseness* into the weight space – only certain token pairs emerge with high weights and are relevant for the remaining part of the considered neural network architecture. In practice, multiple such heads can be concatenated and fed into the succeeding feed-forward layer. The application of softmax has been shown to emphasize only particular parts of the parameter space, thereby making the neural network more focused.

The positional encoding, as discussed in Vaswani et al. [59], represents a matrix that encodes individual positions in a matrix of the same dimensionality as the one holding the information on sequences (input embedding). The positional encoding was introduced to account for *word order*. Here, relative distances between different tokens are taken into account by incorporating the position-related signal into a given token representation.

While there are, in principle, many different ways of how attention networks can be extended with the Bayesian approach, we propose to use the well-established MCD.

3.2 Monte Carlo Dropout for Attention Networks

In our proposal, called Bayesian Attention Networks (BAN), we use MCD within attention networks but contrary to the original dropout setting, the dropout layers are active also during the prediction phase. In this way, the predictions are not deterministic but are sampled from the *learned* distribution, thereby forming an ensemble of predictions. The obtained distribution can be, for example, inspected for higher moment properties and can offer additional information on the uncertainty of a given prediction. During the prediction phase, the dropout layers are activated again and the output of a proportion of randomly selected neurons in those layers is set to zero. A forward pass on such partially activated architecture is repeated for a fixed number of samples, every time dropping different randomly selected neurons. The results of different passes can be combined to obtain the final prediction, or further inspected as a probability distribution.

3.3 Monte Carlo Dropout for BERT

MCD was used in the BERT model in the same way as in BAN. MCD can provide multiple predictions of a neural network during the test time, as long as the dropout was used during the training phase [19]. Training of neural networks with the dropout distributes the captured information across the network. During the prediction, such a trained neural network is robust. Using the dropout principle, a new prediction is possible in each forward pass. A sufficiently large set of such predictions can be used to estimate the prediction reliability. The BERT model is trained with 10% of dropout in all of the layers by default, and thus allows for multiple predictions using the described principle. We call this model 'MCD BERT'. A limitation of this approach is that a single dropout rate of 10% is used during training, while other dropout probabilities might be more suitable for reliability estimation. We leave this analysis for further work.

4 Calibration of Probabilistic Classifiers

The quality of reliability scores returned by probabilistic classifiers (such as BAN and MCD BERT) is assessed with calibration measures. A classifier is calibrated if its output scores are close to actual probabilities in a sense that a class predicted with the score p is correct with the actual probability p, i.e. in $p \cdot 100$ percent of cases. Without special calibration approaches, most neural networks are overconfident and overestimate their probabilities. The calibration of a model can be visualized using a calibration plot where the model's prediction accuracy (true probabilities) is plotted against the predicted probabilities (i.e. outputs scores). The perfect calibration manifests itself as a diagonal in the calibration plot (see an example of a calibration plot in Fig. 6).

Since classifiers are typically not perfectly calibrated, we investigated different methods to improve the calibration of used neural networks. We compared several existing calibration methods with a novel approach that combines existing techniques with a method for threshold adaptation. In Section 4.1, we describe the existing calibration methods, followed by the proposed threshold adaptation in Section 4.2.

4.1 Existing Calibration Methods

We first formally describe how to obtain calibrated predictions from the reliability scores. Let (X, Y) be the input space, where X represents the set of predictive variables, and Y is the binary class variable (either 0 or 1). Let f be the predictor (e.g., neural network) with $f(X) = (\hat{Y}, \hat{P})$, where $\hat{Y}$ is the binary class prediction, and $\hat{P}$ is its associated confidence score or probability score of correct prediction. The calibration of the model f is expressed as:

$$P(\hat{Y} = Y | \hat{P} = \hat{p}) = p, \tag{1}$$

where $\hat{p}$ is the prediction score from $[0, 1]$ interval, obtained from the predictor f. We interpret this score as the probability of a specific outcome, assigned by the model f. Probability p is the model's confidence or true probability that model f predicts correctly. If a model predicts a certain outcome with a high probability, it is desirable that the confidence of this prediction being correct is also high. In the ideal case of perfect calibration $\hat{p} = p$.

Based on Equation (1), there are two ways to reduce the calibration error: either to obtain calibrated predictions $\hat{p}$ or to manipulate the prediction threshold in such a way that the predicted outcome $\hat{Y}$ is better calibrated. To assess the quality of the produced reliability scores, we compare them to results of two calibration methods, Platt's method and Isotonic regression.

Platt's method [50] learns two scalar parameters $a, b \in \mathbb{R}$ in such way that the prediction $\hat{q} = \sigma(a\hat{p} + b)$ presents a calibrated probability of predicted score $\hat{p}$, and σ is the sigmoid function. To find good values of a and b, typically a separate calibration dataset is used. The isotonic regression is a non-parametric form of regression in which we assume that the function is chosen from the class of all isotonic (i.e., non-decreasing) functions [67]. Given the predictions from our classifier $\hat{p}$, and the true target y, the calibrated prediction returned by the isotonic regression is:

$$\hat{q} = m(\hat{p}) + \epsilon$$

where m is a non-decreasing function.

4.2 Adaptive Threshold

We explored the adaptive threshold (AT), which we apply to classification with BANs. During learning, after each weight update phase, we assess the performance of BAN. For each instance in the validation set, we do multiple forward passes with unfrozen dropout layers and store the average of the returned scores as the probability estimate. Once the probability estimates for the validation set are collected, we test several decision thresholds and determine the predictions of each instances. The best-performing threshold w.r.t. a given performance metric (in our case the classification accuracy), is stored together with its performance and weights of the neural network. The obtained performance estimate can also be used for early stopping in the learning phase. When we apply the model to new instances, we use the best threshold from the training phase (instead of the default value of 0.5). The purpose of AT is to automatically find the threshold with the best performance. To summarize, we employ the following procedure:

1. During the training and after each weight update, we generate the probability distribution with MCD. The mean of the distribution is considered the probability score of a given instance being assigned to the positive class.
2. Using the validation set, we test a range of possible thresholds that determine the instances' labels. We tested the threshold range between 0.1 and 0.9 in increments of 0.001.

3. If the accuracy obtained by the default threshold (0.5) was improved by any other threshold, we stored both the current parameter set and the threshold value used to obtain the improved performance on the validation set.
4. The weights of the best performing model and the matching threshold are returned as the final prediction model.

5 Evaluation Settings

In this section, we present the evaluation settings, and in Section 6, we report the results. Starting with Section 5.1, we describe the used hate speech datasets, followed by he affective dimensions of the Hourglass of Emotions method in Section 5.2. The implementation details of the used prediction models are presented in Section 5.3. In Section 5.4, we present the evaluation measures for the predictive performance, and in Section 5.5, the measures used in the evaluation of calibration.

5.1 Hate Speech Datasets

To test the proposed methodology in the multilingual context, we used hate speech datasets in three languages, English, Croatian, and Slovene. The summary of datasets is available in Table 1.

1. The **English** dataset[3] is extracted from hate speech and offensive language detection study of Davidson et al. [14]. The subset of data we used consists of 5,000 tweets. We took 1,430 tweets labeled as the hate speech and randomly sampled 3,670 tweets from the remaining 23,353 tweets.
2. The **Croatian** dataset was provided by the Styria media company within the EMBEDDIA project[4]. The texts consists of user comments on the news portal Večernji list[5]. The original dataset consists of 9,646,634 comments from which we selected 8,422 comments. 50% of instances were labeled as the hate speech by human moderators, and the other half was chosen randomly from non-problematic comments.
3. The **Slovene** dataset was produced in the Slovenian national project FRENK[6]. The text dataset used in the experiment is a combination of two different studies of Facebook comments [30]. The first group of comments was collected on LGBT homophobia topics, while the second on anti-migrants posts. In our final dataset, we used all of the 2,182 hate speech comments, and the same number of non-hate speech comments were randomly sampled.

[3] https://github.com/t-davidson/hate-speech-and-offensive-language
[4] http://embeddia.eu
[5] https://www.vecernji.hr
[6] http://nl.ijs.si/frenk/ (Research on Inappropriate Electronic Communication)

Table 1: Characteristics of the datasets used in the experiments.

Dataset	type	Size	Hate	Non-hate	LSTM embeddings
English	tweets	5000	1430	3670	sentence
Croatian	news comments	8422	4211	4211	fastText
Slovene	Facebook comments	4364	2182	2182	fastText

5.2 The Hourglass of Emotions Affective Dimensions

To test if emotional information extracted from text can complement the information extracted by BERT models, we used the English tweets dataset and affective dimensions obtained with two versions of the Hourglass of Emotions model; the affective dimensions of the original model can be extracted using the *SenticNet 4* framework [9], and the affective dimensions of its revision [58] are available in the *SenticNet 6* framework.

5.2.1 SenticNet 4

We used the SenticPhrase interface to obtain the original Hourglass of Emotions affective dimensions from the SenticNet 4 framework [10]. For each sentence, we extracted four affective dimensions (pleasantness, attention, sensitivity, and aptitude). Within SenticNet 4, verb and noun concepts are linked to primitives, and in this way, most concept inflections can be captured by the knowledge base verb concepts. The implementation is freely accessible via Python API (Application Programming Interface) in the Python sentic package[7].

To gain a better understanding of the four affective dimensions, Cambria et al. [8] presented the following example:

1. The user is happy with the service provided (pleasantness).
2. The user is interested in the information supplied (attention).
3. The user is comfortable with the interface (sensitivity).
4. The user is disposed to use the application (aptitude).

The hate speech texts usually express unhappiness with the current situation and unwillingness to hear or consider different opinions. Hence, the nature of the hate speech is opposite to the nature of pleasantness and aptitude, while it can be correlated with the attention.

The distributions of the affective dimensions for English tweets, separately for non-hate speech and hate speech instances, are shown in Fig. 2. While the distributions are different among the variables, the differences between the hate speech and non-hate speech distributions are not pronounced. This indicates that these variable are not strong indicators of hate speech if used independently, but might still be useful in combination with textual features extracted by neural networks.

[7] `https://pypi.org/project/sentic/`

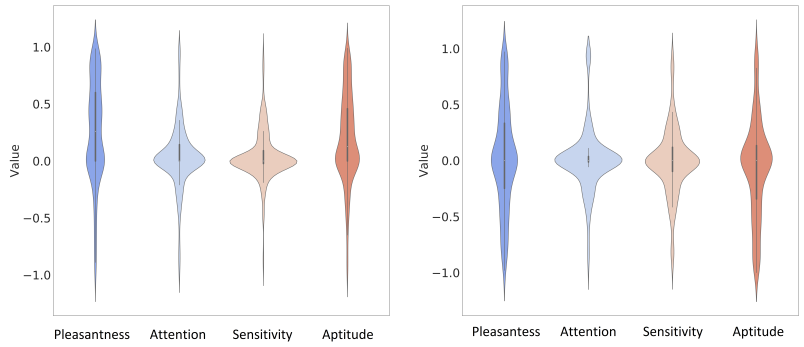


Fig. 2: Distributions of the four affective dimensions from the original Hourglass of Emotions model, obtained from the SenticNet 4 framework for the dataset of English tweets. Left-hand side shows non-hate speech tweets and right-hand side shows hate speech tweets.

5.2.2 SenticNet 6

The revisited Hourglass of Emotions model [58] is based on empirical evidence obtained in the context of sentiment analysis. Each of the four proposed baseline affective dimensions gives positive and negative perspective of one emotion:

1. Introspection - the joy versus sadness;
2. Temper - the calmness versus anger;
3. Attitude - the pleasantness versus disgust, and
4. Sensitivity - the eagerness versus fear.

The dataset of affective dimensions was obtained using the *senticnet* Python library [8]. We used the publicly available word level API to obtain the affective dimension values for each token separately. We averaged the affective dimension and polarity values on the level of each tweet/comment.

We show the distributions of these new dimensions for English tweets in Fig. 3. Similarly to SenticNet 4 framework, the distributions between the hate speech and non-hate speech tweets are similar.

5.3 Implementation of Prediction Models

We used three types of neural network architectures. As a baseline, we used MCD LSTM networks [42], which include reliability information obtained with MCD. We compared that model with newly proposed BAN and MCD BERT. As shown in the right-most column of Table 1, the input to MCD LSTM are pre-trained word embeddings: sentence encoder for English [12], and fastText

[8] `https://pypi.org/project/senticnet/`

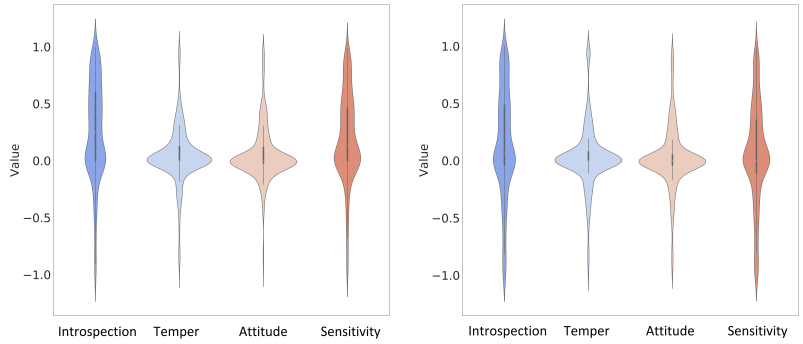


Fig. 3: Distributions of the four affective dimensions from the revisited Hourglass of Emotions model, obtained from the SenticNet 6 framework for the dataset of English tweets. Left-hand side shows non-hate speech tweets and right-hand side shows hate speech tweets.

embeddings[9] for Slovene and Croatian. For the implementation of BAN, we used the Keras tokenizer[10], and for MCD BERT, we used the BERT's tokenizer.

We implemented the proposed BANs[11] and MCD BERT[12] with the PyTorch library. The main hyper-parameters of the BAN architecture are the number of attention heads and the number of attention layers. The adaptive classification threshold (described in Section 4.2) is computed every time we evaluate the performance on the validation set. When a network makes a prediction, we deactivate all layers except the dropout layers. In this way, we maintain the variance of predictions. Each final prediction consists of a set of results obtained by several forward passes.

Other parameters are set as follows. We use the Adamax optimizer [26], a variant of Adam based on infinity norm, and binary cross-entropy loss function. To automatically stop training, we use the stopping step of 10 – if after 10 optimization steps the performance on the validation set is not improved, the training stops.

We explored the following hyperparameter tuning space: the validation percentage (size of the validation set) was varied between 5% and 10%. The rationale for testing different validation set sizes are relatively small datasets, therefore it is difficult to strike a good balance between the training and validation set. Given enough data, the validation set shall be on the upper margin. The number of epochs was either 30 or 100, the number of hidden layers and attention heads was 1 or 2. The maximum padding of the input

[9] `https://fasttext.cc`
[10] `https://keras.io/preprocessing/text/`
[11] `https://github.com/KristianMiok/BAN`
[12] `https://github.com/KristianMiok/Bayesian-BERT`

sequences was either 48, 32, or 64. The learning rate was either 0.001 or 0.0005, and AT was either enabled or disabled.

MCD LSTM networks consist of an embedding layer, LSTM layer, and a fully connected layer within the word2vec [39] and ELMo [49] embeddings. To obtain the best architectures for the LSTM and MCD LSTM models, we tested different number of units, batch sizes, dropout rates, etc.

For BERT, we used the BERT base model in English and the multilingual BERT variant for Croatian and Slovene. We used the HuggingFace implementation[13]. To combine the information from the MCD BERT and SenticNet, we generated 1000 MCD BERT predictions for each instance. We merged them with the four Sentic variables, described in Section 5.2, thus obtaining 1004 variables. This data was passed as an input to the SVM model. The process used 5-fold cross-validation.

5.4 Prediction Performance Evaluation Measures

Depending on the purpose of the prediction model, we might optimize different evaluation measures, such as classification accuracy, precision, recall, or F_1 score. In the hate speech detection, we want to avoid false accusations of hate speech. For that aim, we maximize precision on the validation set during training. As this could negatively affect other measure, we alter the decision threshold to achieve good precision vs. accuracy balance. In Figure 4, we present the accuracy-precision trade-off.

5.5 Calibration Quality Measures

To measure the quality of computed calibration scores, we use the expected calibration error (ECE) [22]. To compute ECE, we split all n predictions into M equally spaced bins $B_1, B_2, \ldots, B_M$, that contain instances with prediction scores in the given bin. We sum the weighted differences between actual prediction accuracies and predicted scores over all the bins and normalizes the result with the number of instances n.

$$ECE = \sum_{m=1}^{M} \frac{|B_m|}{n} |\text{accuracy}(B_m) - \text{score}(B_m)| \quad (2)$$

This measure produces lower scores for better calibrated models (lower calibration error).

6 Results

In this section, we present results of five sets of experiments. In Section 6.1, we report calibration of different prediction models, and in Section 6.2, their

[13] `https://huggingface.co/transformers/model_doc/bert.html`

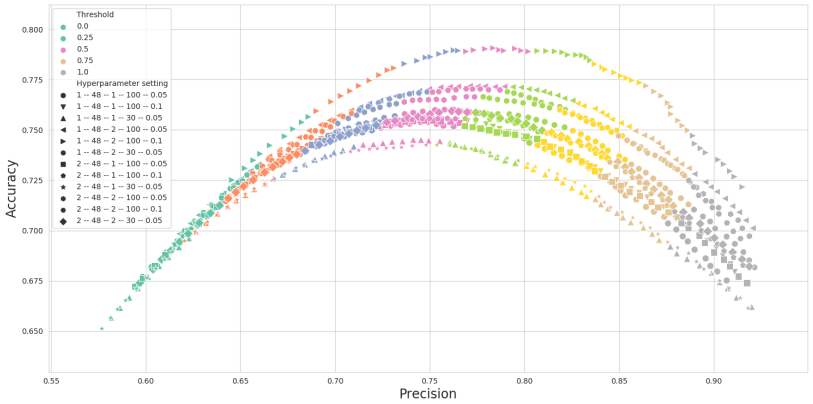


Fig. 4: Trade-off between precision and accuracy across various hyper-parameters settings of BAN model. Each curve shows one set of hyper-parameters, each color depicts one decision threshold (0, 0.25, 0.5, 0.75, or 1.0). The hyper-parameters contain the number of heads, max padding, number of layers, number of epochs, and validation set ratio.

prediction performance. The comparison between the reliability of BERT and MCD BERT is presented in Section 6.3, while the impact of sentic features is discussed in Section 6.4. Finally, we present different visualizations of models' uncertainty in Section 6.5.

6.1 Calibration of BAN and BERT

Figure 5 shows how calibration of prediction scores changes during the training of BAN. The red line represents the performance of the fully trained network. It is apparent that an additional calibration is necessary – as the perfect calibration corresponds to the dotted line. Surprisingly, some of the training iterations show better calibrated scores. This is the motivation for AT, presented in Section 4.2.

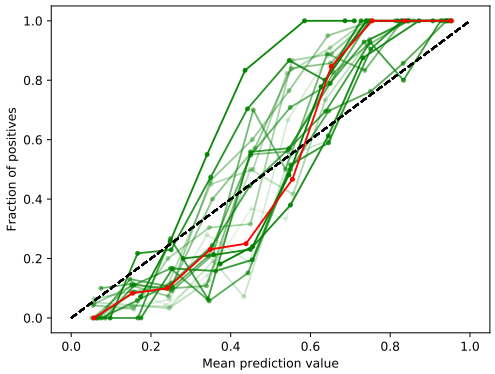


Fig. 5: Calibration plot for the BAN English model after each epoch (green) based on the validation set and the best performing architecture. The transparency of the green calibrations lines decreases with the number of epochs (i.e. initial stages are the most transparent). The final calibration is in red and the dotted line shows the perfect calibration.

In Tables 2, 3, and 4, the calibration results for different calibration settings on the BAN are presented: no calibration, isotonic regression, and Platt's method. Each calibration is either combined with AT or not. For all three languages, both calibration methods improve the ECE score, and Platt's method produces the best calibration scores. The AT slightly improves the ECE score for the uncalibrated (raw) results. This is especially true for the Slovene comments where the ECE score was reduced from 0.794 to 0.621. We can conclude that the calibration using AT heuristics might not be beneficial when used in combination with the established calibration techniques (isotonic regression and Platt's method) but used exclusively.

To compare the calibration of MCD BERT with different BAN calibrations, we plotted their ECE scores in Figure 6. It can be observed that calibration methods substantially improve the BAN score. However, the MCD BERT model is better calibrated even without the usage of an explicit calibration methods.

6.2 Prediction Performance

We compare the predictive performance of four neural network architectures in Table 5. MCD LSTM and BERT serve as the baselines for comparison with the proposed BAN and MCD BERT. The MCD BERT model provides the best results for all three languages. BERT models are pre-trained on large amounts of text, which makes a significant difference compared to LSTM and BAN.

Table 2: The calibration scores of BAN with different calibration approaches on the English tweets dataset. We present average classification accuracy and F1 score with their standard deviations, computed using 5-fold cross-validation.

Calibration	AT	Accuracy	F1	ECE
Raw	False	0.83 [0.02]	0.82 [0.03]	0.547
Raw	True	0.83 [0.01]	**0.83** [0.04]	0.539
Isotonic	False	**0.84** [0.01]	0.82 [0.01]	0.230
Isotonic	True	0.83 [0.01]	0.82 [0.02]	0.234
Platt's	False	**0.84** [0.02]	0.82 [0.02]	**0.225**
Platt's	True	0.83 [0.01]	0.82 [0.01]	0.232

Table 3: The calibration scores of BAN with different calibration approaches on the Croatian user news comments dataset.

Calibration	AT	Accuracy	F1	ECE
Raw	False	0.61 [0.02]	0.47 [0.03]	0.681
Raw	True	**0.62** [0.02]	**0.50** [0.04]	0.663
Isotonic	False	0.60 [0.01]	0.49 [0.04]	0.206
Isotonic	True	0.61 [0.01]	**0.50** [0.03]	0.206
Platt's	False	0.61 [0.02]	0.48 [0.02]	0.198
Platt's	True	**0.62** [0.02]	0.49 [0.02]	**0.197**

Table 4: The calibration scores of BAN with different calibration approaches on the Slovene Facebook comments dataset.

Calibration	AT	Accuracy	F1	ECE
Raw	False	**0.59** [0.01]	0.33 [0.05]	0.794
Raw	True	**0.59** [0.02]	0.48 [0.05]	0.621
Isotonic	False	0.58 [0.02]	0.48 [0.03]	0.212
Isotonic	True	0.58 [0.02]	**0.49** [0.03]	0.213
Platt's	False	0.58 [0.03]	0.475 [0.02]	0.206
Platt's	True	**0.59** [0.02]	0.47 [0.04]	**0.204**

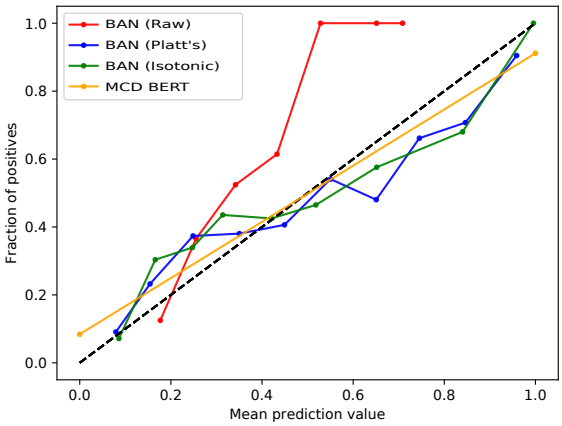


Fig. 6: Calibration plots based on English test set performance for MCD BERT and BAN with different calibration algorithms.

MCD BERT is slightly better than BERT due to its better performance for the instances where BERT is uncertain. Here, multiple predictions reduce the prediction variance. MCD LSTM is more stable than BAN (see the standard deviation of F_1 score in (see the standard deviation of F_1 scores in Table 5). We attribute this to the larger number of parameters in BAN and insufficient number of training instances. BERT and MCD BERT models compensate for this problem with large scale pre-training.

Table 5: Predictive performance of compared models. We present the average classification accuracy and F_1 score with their standard deviations (in brackets), computed using 5-fold cross-validation. The best accuracy for each language is typeset in bold.

	English Tweets		**Croatian Comments**		**Slovene Comments**	
Model	**Accuracy**	**F1**	**Accuracy**	**F1**	**Accuracy**	**F1**
MCD LSTM	81.0 [1.2]	81.9 [1.3]	63.7 [1.0]	51.0 [3.3]	55.3 [0.69]	43.13 [0.8]
BAN	83.3 [1.7]	81.6 [3.4]	61.4 [2.0]	38.1 [8.6]	57.4 [1.7]	35.1 [6.3]
BERT	90.9 [0.7]	90.0 [0.7]	70.8 [1.0]	61.2 [1.5]	66.4 [5.0]	67.8 [2.5]
MCD BERT	**91.4 [0.7]**	**90.4 [0.8]**	**71.5 [1.2]**	**62.9 [1.7]**	**68.4 [1.9]**	**68.6 [1.6]**

6.3 Reliability of BERT and MCD BERT

As established in Section 6.1, BERT models are already well-calibrated. In this section, we test if the proposed MCD BERT extension is useful beyond the advantage in predictive performance, and analyze the ability of MCD BERT to detect problematic predictions. For each classifier (BERT and MCD BERT), we split the tested instances into two groups, *uncertain* and *certain*, based on the computed prediction scores. As BERT and MCD BERT return most of the predictions close to 0 or 1, we used the following criteria for the certainty of prediction scores. For MCD BERT, the tested instance is declared *uncertain* if the variance computed on 1000 dropout predictions is greater then 0.1, otherwise it is declared *certain*. As BERT returns a single prediction score, we have chosen the same number of *uncertain* instances as for MCD BERT, based on the criterion that their prediction scores are farthest away from 0 or 1, i.e. they are least certain to be either hate speech or not.

In Table 6, we show the number of predictions where classifiers are correct/incorrect separately for instances with certain/uncertain prediction for each of the three languages. The ratio of incorrectly to correctly classified instances is significantly different between the *certain* and *uncertain* group, which is a strong indication that both BERT and MCD BERT correctly recognize uncertain predictions. This ratio is also much larger for MCD BERT than for BERT for the English and Croatian dataset, which testifies that the reliability of MCD BERT predictions is better. The ratio is similar for the Slovene dataset, where BERT also has a good ratio.

Using the Chi-square statistical test, we assessed the difference in correct/incorrect classifications between the *certain* and *uncertain* group. For the English dataset, this difference is highly significant for both BERT and MCD BERT (p =1.384e-11 and 2.2e-16, respectively). For the Croatian dataset, the p-values are 1 and 8.348e-16, meaning that we cannot rely on BERT scores to detect uncertain classifications, while the distribution returned by the MCD

Table 6: The number and ratio of predictions where classifiers are correct/incorrect is very different for instances where BERT and MCD BERT are certain/uncertain. We use three datasets, English (ENG), Croatian (CRO), and Slovenian (SLO).

		BERT		MCD BERT	
Language	**Correct**	**Certain**	**Uncertain**	**Certain**	**Uncertain**
ENG	**Yes**	880	31	891	24
	No	71	18	62	23
	N/Y Ratio	0.08	0.58	0.06	0.95
CRO	**Yes**	1176	35	1053	152
	No	461	14	336	139
	N/Y Ratio	0.39	0.40	0.31	0.91
SLO	**Yes**	576	28	537	55
	No	241	27	229	51
	N/Y Ratio	0.42	0.96	0.42	0.92

BERT is very informative. The p-values for the BERT and MCD BERT on Slovene are 0.0037 and 0.0002, respectively. Again, MCD BERT is much better in detecting unreliable classifications.

The observed difference in assessment of reliability can have important practical consequences. For example, if we are faced with the re-annotation task to improve the quality of predictions, MCD BERT would choose much better borderline instances compared to BERT.

6.4 Combining Emotional Information with MCD BERT

As the experiments in Section 6.2 show, MCD BERT is superior to other tested models on the hate speech detection task. In this section, we test if additional emotional information obtained from the SenticNet framework can complement the information about the hate speech extracted by the MCD BERT model and further improve its performance. We merge the affective dimensions computed based on SenticNet 4 and SenticNet 6 with the output vector of MCD BERT predictions, described in Section 3.3. Additionally, we investigate if the emotional information can help in the interpretation of trained hate speech detectors.

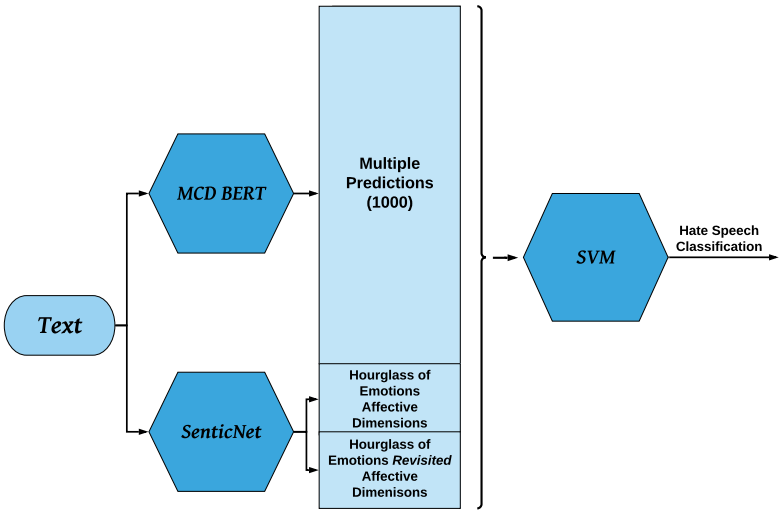


Fig. 7: A diagram of merging MCD BERT predictions with the emotional information based on SenticNet 4 and SenticNet 6 frameworks. The concatenated vector is an input to the final SVM classifier that predicts the hate speech.

For the evaluation, we use 5-fold cross-validation. In each iteration, we combine the predictions from MCD BERT (1000 of them, sorted in ascending order) with the affective dimensions from the original and revised variant of the Hourglass of Emotions models as depicted in Figure 7. We obtain four affective dimensions from the original Hourglass of Emotions model (pleasantness, aptitude, sensitivity, and attention), and four from the revisited model (introspection, sensitivity, temper, and attitude). Using the dataset obtained in this way, we train the SVM model to predict the hate speech. According to the results in Table 7, the additional information does not improve the hate speech detection. The same conclusions can be drawn from Figure 8, where we plot the scores assigned to the used features by the random forest algorithm [5]. This learning algorithm can detect feature dependencies that affect the prediction variable. Thus, the results show that SVM and random forest cannot detect any pronounced interactions between affective dimensions and MCD BERT predictions that would impact the hate speech classification.

The results show that introducing knowledge regarding emotional content after the predictions are done can not improve the performance. However, according to the authors of the Hourglass of Emotions revisited model [58], the full sentence model introduced in SenticNet 6 [11] can provide superior text classification results on problems involving emotions. Thus, the layers that can capture emotional information from the text should be build within the prediction model architecture. Introducing uncertainty component in such architecture remains an interesting direction for further research.

Table 7: Predictive performance of the MCD BERT model and the SVM model trained on the output features of MCD BERT and affective dimensions from the two Hourglass of Emotions models for the English tweets dataset.

Model	Accuracy	F1
MCD BERT	**91.4** [**0.7**]	**90.4** [**0.8**]
MCD BERT + ***SenticNet 4*** + ***SenticNet 6***	**91.4** [**0.5**]	**90.5** [**0.9**]

To better understand the emotions involved in hate speech problem, we further investigated the relation between the affective dimensions of the two Hourglass of Emotions models (original and revisited) and the hate speech prediction probabilities of MCD BERT, separately for the non-hate speech and hate speech English tweets.

The top line of Figure 9 shows results for the affective dimensions of the original Hourglass of Emotions models (pleasantness, attention, sensitivity, and aptitude). The top parts of graphs show that linear regression lines (in orange) for the hate speech are almost horizontal, so there is no significant correlation between the predicted probability of hate speech obtained with MCD BERT and affective dimensions. In contrast, the correlation between the predicted probability and affective dimensions for the non-hate speech tweets is significant, as the blue regression lines at the bottom parts of the graphs

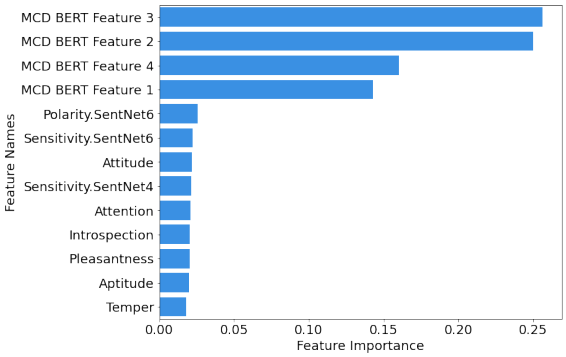


Fig. 8: Feature importance scores according to the random forest algorithm. We show scores of 8 affective dimensions extracted from the SenticNet 4 and SenticNet 6 frameworks, as well as five most important attributes generated by the MCD BERT model.

in the top row show. Both *attention* and *sensitivity* have positive correlation with the hate speech prediction probability. This is in accordance with the conclusions of the original Hourglass of Emotions model that high *attention* and *sensitivity* lead to aggressiveness (Figure 5 in [9]).

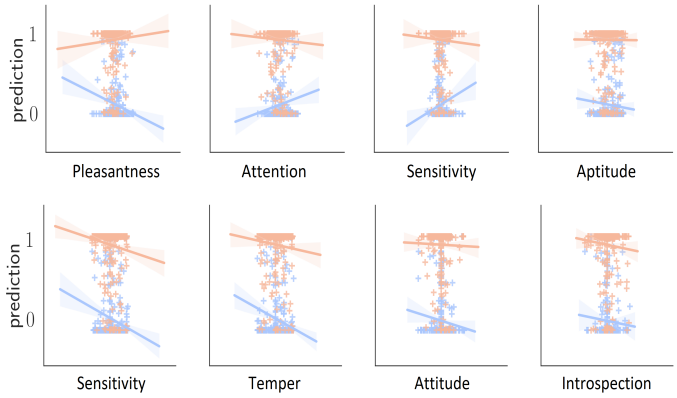


Fig. 9: Relationship between prediction probability of MCD BERT and the Hourglass of Emotions affective dimensions. Original affective dimensions are shown in the top line, while revisited dimensions are shown in the bottom line of graphs.

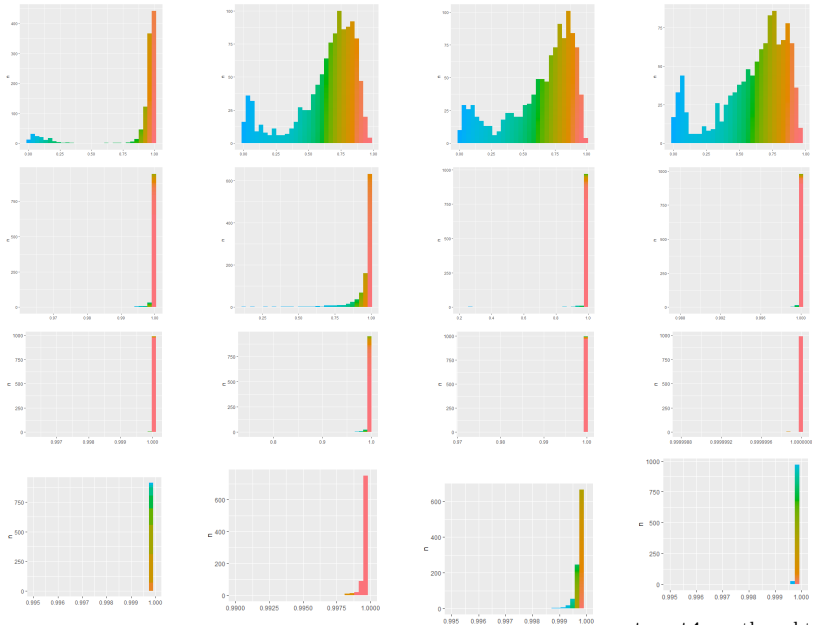
The bottom line of Figure 9 shows the affective dimensions of the revisited Hourglass of Emotions model (sensitivity, temper, attitude and introspection). These affective dimensions are all negatively correlated with the non-hate speech. It can be observed that there is also a slight negative correlation between the affective dimensions and hate speech probabilities, especially for sensitivity and temper. Thus, tweets that contain dominantly positive emotions have a low probability of being hate speech which is in accordance with the results presented by Susanto et al. [58].

6.5 Visualization of Uncertainty

Obtaining multiple predictions for a specific instance can improve understanding of the final prediction. We used the mean of the distribution to estimate the probability. The variance informs us about the spread and certainty of predictions. We can inspect the actual distribution of prediction scores with histogram plots, as illustrated in Figure 10 for a few correctly classified instances from the English dataset, and on Figure 11 for a few misclassified instances. We analyze distributions produced by the MCD LSTM baseline, BAN with 10% and 30% dropout, and MCD BERT.

Histograms in Figures 10 and 11 visually display the prediction certainty for the specific instances. We notice that MCD BERT's predictions are always close to 0 or 1, especially when the model seems certain of the prediction. BAN with 10% dropout provides a similar spread of values as MCD BERT. This is expected as BERT is also pre-trained with 10% dropout. However, 30 % of dropout in BAN results in a much larger spread of predictions for instances where BAN is uncertain. Note that the results of MCD BERT are concentrated in a much narrower interval compared to MCD LSTM and BAN.

While visualizations of prediction distributions for individual instance (see Figures 10 and 11) are useful in the assessment of their prediction reliability, we also aggregate results over multiple instances to understand more general reliability phenomena. Following [42], we visualize the embeddings of the prediction distributions. The idea of this visualization is to detect and identify clusters of certain and uncertain classifications. First, we obtain many predictions (1,000 in our experiments) for each instance. The space of prediction distributions across instances is embedded into two dimensions by the Uniform Manifold Projections method (UMP) [37]. In this way, we obtain a two-dimensional space corresponding to the initial 1,000 dimensional space of prediction distributions. Next, we use the Gaussian kernel estimation to identify equivalent regions and connect them with closed curves. Finally, the shapes and sizes of individual predictions are chosen based on their classification error and certainty of predictions. The goal of this visualization is to discover structures within the space of probability distributions, possibly offering insights into the drawbacks and limitations of the analyzed classifier. The resulting visualizations are shown in Figures 12 and 13. In Figure 12, the plot displays the position of certain and



tweet1: @user you might be a libtard if... #libtard #sjw #liberal #politics

tweet2: #paid #kkk to #fabricate #stories to push the producers #narrative #cancel

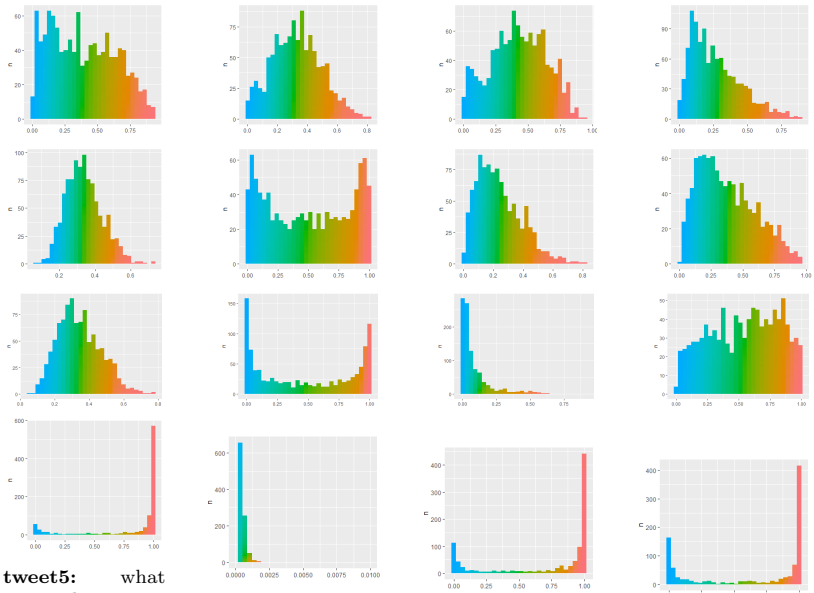
tweet3: carl paladino, trump ally, wishes obama dead of mad cow disease

tweet4: thought factory: left-right polarisation! #trump #uselections2016 #leadership #politics #brexit #blm

Fig. 10: Distributions of prediction scores for a few *correctly* classified English instances. We show histograms for MCD LSTM (first row), BAN with 30% dropout (second row), BAN with 10% dropout (third row), and MCD BERT (fourth row). Each tweet is shown in a separate column.

uncertain test set instances in the embedded space of distributions, while in Figure 13 the differences are based on the mean of predicted probability scores.

In both Figures, 12 and 13, the probability space is distinctly separated into two components, indicating that there are predictions for which the neural network is certain (and were correctly classified). However, for some predictions, especially non hate speech instances, the model is less certain (albeit still correct). The two visualizations demonstrate how the probability space is split into distinct components for a trained neural network. The visualizations also shows problematic predictions, allowing their identification and potentially facilitating the debugging process for developers (e.g., an inspection of convergence).



tweet5: what over-rode journalists' integrity was greed and ambition, along with a total absence of courage (true label = hate speech)

tweet6: there is a future nina and right now you are pretty much throwing yours away. (true label = non-hate speech)

tweet7: @user @user someone really respects eu guidelines!!! (true label = non-hate speech)

tweet8: the scars left by us waime camps mt @user (true label = hate speech)

Fig. 11: Distributions of prediction scores for a few *incorrectly* classified English instances. We show histograms for MCD LSTM (first row), BAN with 30% dropout (second row), BAN with 10% dropout (third row), and MCD BERT (fourth row). Each tweet is shown in a separate column.

7 Conclusions and Future Work

In real world scenarios, an automatic detection of hate speech requires high precision and reliable decisions. Wrong classifications can lower the level of democratic debate and damage freedom of speech. In technological terms, NLP is witnessing a switch from RNNs with pre-trained word embeddings (such as LSTM with fastText) to large pre-trained transformer models (such as BERT).

We proposed to use the MCD in the attention layers of transformer neural networks, and to unfreeze dropout layers also during the prediction phase. This resulted in two new architectures, BAN and MCD BERT. The BAN models are transformer networks trained from scratch, using dropout in both training and prediction phase. MCD BERT uses pretrained BERT model and uses dropout

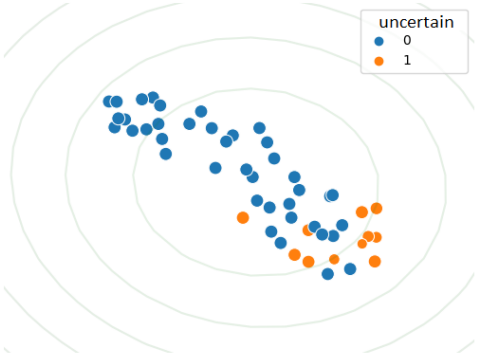


Fig. 12: Visualization of 100 test tweets projected into two dimensional space by the UMP method. Tweets whose classifications seem certain are colored in blue while tweets with uncertain classification are shown in orange. We can observe clustering of uncertain tweets.

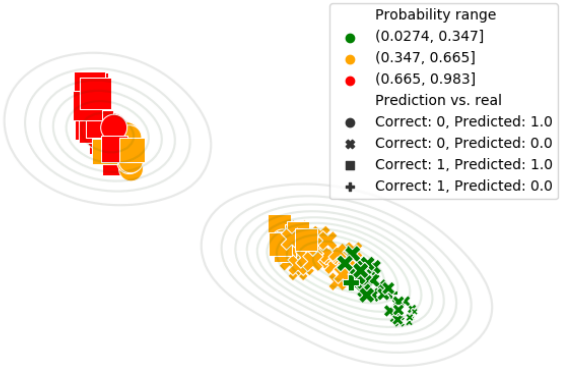


Fig. 13: Visualization of the probability space for 100 tweets from the test set. The instances are colored green, yellow, or red, depending on the mean probability of the 1000 predictions. Predictions with high confidence form an isolated part in the probability space.

during fine-tuning and prediction phase. We have shown that these approaches are useful for estimation of prediction uncertainty. MCD BERT significantly improves the prediction performance in the hate speech detection task. Its pre-training extracts useful information about the language use that can be successfully exploited in the fine-tuning to a specific problem. BANs, trained

from scratch, are not competitive with this. We also empirically investigated the calibration of BAN and MCD BERT. The results show that MCD BERT is much better calibrated than BAN.

Multiple predictions obtained from MCD BERT not only produce better predictive performance compared to BERT, but also provide better reliability information. The visualizations based on them enable detection of less certain decisions and can help moderators or annotators to focus on uncertain instances.

In line with the recent research showing that the affective information available in the SenticNet 6 framework provides favorable results in the sentiment analysis [58], we tested this information on the hate speech detection task. We combined affective dimensions from the original and revisited Hourglass of Emotions models with predictions generated by the MCD BERT model. While our results do not show any improvement in predictive performance, we believe inclusion of affective information should be incorporated within the prediction model together with possibility of obtaining prediction uncertainty. Thus, we see an opportunity for further work in this area by introducing BERT-based uncertainty estimated into full sentence models from the SenticNet 6 framework. Nevertheless, the predictions of the MCD BERT model confirm the findings of the Hourglass of Emotions model. The affective dimensions of the Hourglass of Emotions model are correlated with the non-hate speech probabilities returned by the MCD BERT, and can potentially explain emotions involved in the hate speech. Breaking down a complex offensive language to fundamental emotions can bring interesting insights into the hate speech problem.

In future work, we aim to adapt other Bayesian approaches, such as SWAG, to transformer networks. Reliability enhanced classifications could be used in many other domains, such as machine translation. One of the tasks where Bayesian text classification can be particularly useful is semi-supervised learning, which iteratively expands an initial small set of manually labeled instances with the most reliably classified instances. Data re-annotation is another example where reliability scores can be of great use. An initial pilot study on Croatian comment filtering showed that human annotators decide mostly based on the observed keywords and lack the time to detect more subtle expressions of offensive content. These circumstances result in low quality of the resulting datasets and demand their reannotation. Using the reliability scores of the proposed MCD BERT, one could significantly reduce the amount of reannotation and focus on genuinely difficult and borderline cases where prediction models may err.

Funding Information Marko Robnik-Šikonja received the financial support from the Slovenian Research Agency through core research programme P6-0411. All the authors except Daniela Zaharie have received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825153 (EMBEDDIA, Cross-Lingual Embeddings for Less-Represented Languages in European News Media).

Compliance with Ethical Standards

Conflict of Interest The authors declare that they have no conflict of interest.

Ethical Approval This article does not contain any studies with human participants or animals performed by any of the authors.

Informed Consent Informed consent was not required as no humans or animals were involved.

References

1. Alorainy W, Burnap P, Liu H, Javed A, Williams ML (2018) Suspended accounts: A source of tweets with disgust and anger emotions for augmenting hate speech data sample. In: International Conference on Machine Learning and Cybernetics (ICMLC), vol 2, pp 581–586
2. Bauwelinck N, Lefever E (2019) Measuring the impact of sentiment for hate speech detection on Twitter. In: The Fifth International Conference on Human and Social Analytics (HUSO), pp 17–22
3. Bleich E (2011) The rise of hate speech and hate crime laws in liberal democracies. Journal of Ethnic and Migration Studies 37(6):917–934
4. Bluche T, Kermorvant C, Louradour J (2015) Where to apply dropout in recurrent neural networks for handwriting recognition? In: 2015 13th International Conference on Document Analysis and Recognition (ICDAR), pp 681–685
5. Breiman L (2001) Random forests. Machine Learning Journal 45:5–32
6. Cambria E (2016) Affective computing and sentiment analysis. IEEE intelligent systems 31(2):102–107
7. Cambria E, Hussain A (2015) Sentic Computing: A Common-Sense-Based Framework for Concept-Level Sentiment Analysis. Springer, Cham, Switzerland
8. Cambria E, Chandra P, Sharma A, Hussain A (2010) Do not feel the trolls. ISWC, Shanghai
9. Cambria E, Livingstone A, Hussain A (2012) The hourglass of emotions. In: Cognitive behavioural systems, Springer, pp 144–157
10. Cambria E, Poria S, Bajpai R, Schuller B (2016) Senticnet 4: A semantic resource for sentiment analysis based on conceptual primitives. In: Proceedings of COLING 2016, the 26th international conference on computational linguistics: Technical papers, pp 2666–2677
11. Cambria E, Li Y, Xing FZ, Poria S, Kwok K (2020) Senticnet 6: Ensemble application of symbolic and subsymbolic ai for sentiment analysis. In: Proceedings of the 29th ACM International Conference on Information & Knowledge Management, pp 105–114
12. Cer D, Yang Y, Kong Sy, Hua N, Limtiaco N, John RS, Constant N, Guajardo-Cespedes M, Yuan S, Tar C, et al. (2018) Universal sentence encoder for English. In: Proceedings of the 2018 Conference on Empirical

Methods in Natural Language Processing: System Demonstrations, pp 169–174
13. Chang WC, Yu HF, Zhong K, Yang Y, Dhillon I (2019) X-BERT: eXtreme multi-label text classification with BERT. arXiv preprint arXiv:190502331
14. Davidson T, Warmsley D, Macy M, Weber I (2017) Automated hate speech detection and the problem of offensive language. In: Proceedings of the 11th International AAAI Conference on Web and Social Media, ICWSM '17, pp 512–515
15. Devlin J, Chang MW, Lee K, Toutanova K (2019) BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pp 4171–4186
16. Du J, Xu R, He Y, Gui L (2017) Stance classification with target-specific neural attention networks. In: Proceedings of International Joint Conferences on Artificial Intelligence, IJCAI 2017, pp 3988–3994
17. Fišer D, Erjavec T, Ljubešić N (2017) Legal framework, dataset and annotation schema for socially unacceptable online discourse practices in Slovene. In: Proceedings of the first workshop on abusive language online, pp 46–51
18. Fortunato M, Blundell C, Vinyals O (2017) Bayesian recurrent neural networks. arXiv preprint arXiv:170402798
19. Gal Y (2016) Uncertainty in deep learning. University of Cambridge 1:3
20. Gal Y, Ghahramani Z (2016) Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In: International Conference on Machine Learning, pp 1050–1059
21. Gal Y, Ghahramani Z (2016) A theoretically grounded application of dropout in recurrent neural networks. In: Advances in Neural Information Processing Systems, pp 1019–1027
22. Guo C, Pleiss G, Sun Y, Weinberger KQ (2017) On calibration of modern neural networks. In: Proceedings of the 34th International Conference on Machine Learning, pp 1321–1330
23. Gururangan S, Dang T, Card D, Smith NA (2019) Variational pretraining for semi-supervised text classification. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pp 5880–5894
24. Izmailov P, Maddox WJ, Kirichenko P, Garipov T, Vetrov D, Wilson AG (2020) Subspace inference for Bayesian deep learning. In: Uncertainty in Artificial Intelligence, PMLR, pp 1169–1179
25. Kant N, Puri R, Yakovenko N, Catanzaro B (2018) Practical text classification with large pre-trained language models. arXiv preprint arXiv:181201207
26. Kingma DP, Ba J (2014) Adam: A method for stochastic optimization. arXiv preprint arXiv:14126980
27. Kocijan K, Košković L, Bajac P (2019) Detecting hate speech online: A case of Croatian. In: International Conference on Automatic Processing of Natural-Language Electronic Texts with NooJ, pp 185–197

28. Lakshminarayanan B, Pritzel A, Blundell C (2017) Simple and scalable predictive uncertainty estimation using deep ensembles. In: Advances in neural information processing systems, pp 6402–6413
29. Ljubešić N, Erjavec T, Fišer D (2018) Datasets of Slovene and Croatian moderated news comments. In: Proceedings of the 2nd Workshop on Abusive Language Online (ALW2), pp 124–131
30. Ljubešić N, Fišer D, Erjavec T (2019) The FRENK datasets of socially unacceptable discourse in Slovene and English. In: International Conference on Text, Speech, and Dialogue, Springer, pp 103–114
31. Ma Y, Peng H, Khan T, Cambria E, Hussain A (2018) Sentic LSTM: A hybrid network for targeted aspect-based sentiment analysis. Cognitive Computation 10(4):639–650
32. Maddox WJ, Izmailov P, Garipov T, Vetrov DP, Wilson AG (2019) A simple baseline for Bayesian uncertainty in deep learning. In: Advances in Neural Information Processing Systems, pp 13132–13143
33. Malmasi S, Zampieri M (2017) Detecting hate speech in social media. In: Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017, Varna, Bulgaria, pp 467–472
34. Marinšek R (2019) Cross-lingual embeddings for hate speech detection in comments. MSc thesis, University of Ljubljana, Faculty of Computer and Information Science
35. Martins R, Gomes M, Almeida JJ, Novais P, Henriques P (2018) Hate speech classification in social media using emotional analysis. In: 7th Brazilian Conference on Intelligent Systems (BRACIS), pp 61–66
36. McAuley J, Targett C, Shi Q, Van Den Hengel A (2015) Image-based recommendations on styles and substitutes. In: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp 43–52
37. McInnes L, Healy J, Melville J (2018) UMAP: Uniform manifold approximation and projection for dimension reduction. arXiv preprint arXiv:180203426
38. Mehdad Y, Tetreault J (2016) Do characters abuse more than words? In: Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue, pp 299–303
39. Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J (2013) Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems, pp 3111–3119
40. Miok K (2018) Estimation of prediction intervals in neural network-based regression models. In: 20th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), pp 463–468
41. Miok K, Nguyen-Doan D, Robnik-Šikonja M, Zaharie D (2019) Multiple imputation for biomedical data using Monte Carlo dropout autoencoders. 2019 E-Health and Bioengineering Conference (EHB) pp 1–4
42. Miok K, Nguyen-Doan D, Škrlj B, Zaharie D, Robnik-Šikonja M (2019) Prediction uncertainty estimation for hate speech classification. In: International Conference on Statistical Language and Speech Processing, Springer,

pp 286–298
43. Miok K, Nguyen-Doan D, Zaharie D, Robnik-Šikonja M (2019) Generating data using Monte Carlo dropout. In: 2019 IEEE 15th International Conference on Intelligent Computer Communication and Processing (ICCP), IEEE, pp 509–515
44. Mozafari M, Farahbakhsh R, Crespi N (2019) A BERT-based transfer learning approach for hate speech detection in online social media. In: International Conference on Complex Networks and Their Applications, Springer, pp 928–940
45. Myshkov P, Julier S (2016) Posterior distribution analysis for Bayesian inference in neural networks. In: Workshop on Bayesian Deep Learning, NIPS 2016, Barcelona, Spain
46. Niculescu-Mizil A, Caruana R (2005) Predicting good probabilities with supervised learning. In: Proceedings of the 22nd international conference on Machine learning, pp 625–632
47. Pamungkas EW, Patti V (2019) Cross-domain and cross-lingual abusive language detection: A hybrid approach with deep learning and a multilingual lexicon. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop, pp 363–370
48. Pappas N, Popescu-Belis A (2017) Multilingual hierarchical attention networks for document classification. In: 8th International Joint Conference on Natural Language Processing, pp 1015–1025
49. Peters M, Neumann M, Iyyer M, Gardner M, Clark C, Lee K, Zettlemoyer L (2018) Deep contextualized word representations. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), pp 2227–2237
50. Platt JC (1999) Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In: Advances in large margin classifiers, MIT Press, pp 61–74
51. Plutchik R (1982) Emotion, a psychoevolutionary synthesis. Harper & Row, New York
52. Rodríguez A, Argueta C, Chen YL (2019) Automatic detection of hate speech on Facebook using sentiment and emotion analysis. In: 2019 International Conference on Artificial Intelligence in Information and Communication (ICAIIC), pp 169–174
53. Rogers A, Kovaleva O, Rumshisky A (2020) A primer in BERTology: What we know about how BERT works. arXiv preprint arXiv:200212327
54. Schmidt A, Wiegand M (2017) A survey on hate speech detection using natural language processing. In: Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media, pp 1–10
55. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research 15(1):1929–1958
56. Stappen L, Brunn F, Schuller B (2020) Cross-lingual zero-and few-shot hate speech detection utilising frozen transformer language models and

AXEL. arXiv preprint arXiv:200413850
57. Sun C, Qiu X, Xu Y, Huang X (2019) How to fine-tune BERT for text classification? In: China National Conference on Chinese Computational Linguistics, pp 194–206
58. Susanto Y, Livingstone AG, Ng BC, Cambria E (2020) The hourglass model revisited. IEEE Intelligent Systems 35(5):96–102
59. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I (2017) Attention is all you need. In: Advances in neural information processing systems, pp 5998–6008
60. Vezjak B (2018) Radical hate speech: The fascination with Hitler and fascism on the Slovenian webosphere. Solsko Polje 29
61. Wang S, Manning C (2013) Fast dropout training. In: International Conference on Machine Learning, pp 118–126
62. Warner W, Hirschberg J (2012) Detecting hate speech on the world wide web. In: Proceedings of the second workshop on language in social media, pp 19–26
63. Waseem Z, Hovy D (2016) Hateful symbols or hateful people? Predictive features for hate speech detection on Twitter. In: Proceedings of the NAACL student research workshop, pp 88–93
64. Wiedemann G, Yimam SM, Biemann C (2020) UHH-LT & LT2 at SemEval-2020 Task 12: Fine-tuning of pre-trained transformer networks for offensive language detection. arXiv preprint arXiv:200411493
65. Xu Y, Qiu X, Zhou L, Huang X (2020) Improving BERT fine-tuning via self-ensemble and self-distillation. arXiv preprint arXiv:200210345
66. Yang Z, Yang D, Dyer C, He X, Smola A, Hovy E (2016) Hierarchical attention networks for document classification. In: Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies, pp 1480–1489
67. Zadrozny B, Elkan C (2002) Transforming classifier scores into accurate multiclass probability estimates. In: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, pp 694–699
68. Zaremba W, Sutskever I, Vinyals O (2014) Recurrent neural network regularization. arXiv preprint arXiv:14092329
69. Zhu L, Laptev N (2017) Deep and confident prediction for time series at Uber. In: IEEE International Conference on Data Mining Workshops (ICDMW), pp 103–110

Appendix C: Bayesian Methods for Semi-supervised Text Annotation

Bayesian Methods for Semi-supervised Text Annotation

Kristian Miok[1,2], **Gregor Pirš**[1] **and Marko Robnik-Šikonja**[1]

[1] University of Ljubljana, Faculty of Computer and Information Science, Slovenia
Email: {gregor.pirs, marko.robnik}@fri.uni-lj.si
[2] West University of Timisoara, Computer Science Department, Romania
Email: kristian.miok@e-uvt.ro

Abstract

Human annotations are an important source of information in the development of natural language understanding approaches. As under the pressure of productivity annotators can assign different labels to a given text, the quality of produced annotations frequently varies. This is especially the case if decisions are difficult, with high cognitive load, requires awareness of broader context, or careful consideration of background knowledge. To alleviate the problem, we propose two semi-supervised methods to guide the annotation process: a Bayesian deep learning model and a Bayesian ensemble method. Using a Bayesian deep learning method, we can discover annotations that cannot be trusted and might require reannotation. A recently proposed Bayesian ensemble method helps us to combine the annotators' labels with predictions of trained models. According to the results obtained from three hate speech detection experiments, the proposed Bayesian methods can improve the annotations and prediction performance of BERT models.

1 Introduction

Recent successful applications of artificial intelligence in various fields, including natural language processing, are often due to long hours of human annotation when preparing datasets for machine learning. The annotation process transfers human knowledge to machine learning models but it is often done under time pressure and with inadequate instructions or with insufficiently trained annotators.Aiming to make the annotation process easier, we study the possibility of designing a data labeling process which requires less human supervision.

In practice, a fairly standard procedure in the annotation quality control is to recheck the labels that are wrongly classified by using several prediction models. As an alternative, Bayesian inference produces a distribution of possible decisions and can improve the selection of instances requiring reannotation (Miok et al., 2020). Most neural networks do not support the assessment of predictive uncertainty. The Bayesian inference framework can be helpful, however, most techniques do not scale well in neural networks with high dimensional parameter space (Izmailov et al., 2019). Various methods were proposed to overcome this problem (Myshkov and Julier, 2016), one of the most efficient being Monte Carlo Dropout (MCD) (Gal and Ghahramani, 2016a). Its idea is to use the dropout mechanism in neural networks as a regularization technique (Srivastava et al., 2014) and interpret it as a Bayesian optimization approach that samples from the approximate posterior distribution.

A common problem in text annotations is that annotators are not always sure about correct labels due to uncertainty in the text (Vincze, 2015; Szarvas et al., 2008). On difficult texts, annotators frequently give ambiguous labels and their annotations can be biased. Instead of asking annotators to label the raw text, it would be easier for them if they were proposed answers accompanied by probabilistic scores from an ensemble of predictive models. Ensemble methods produce robust models that frequently provide significantly better predictions than individual models. The key strength of ensembles is that they can overcome errors and shortcomings of individual ensemble members. However, diversity in combining different predictions and reliability of individual predictions need to be better understood and

evaluated (Zhou, 2012). A recently published ensemble method Multivariate Normal Mixture Conditional Likelihood Model (MM) (Pirš and Štrumbelj, 2019) tries to understand the predictors on the distributional level and use Bayesian inference to combine them. In this work, we evaluate MM's performance when combining predictive models on the hate speech detection task. We show that our methodology can serve as a helpful tool in the data annotation process.

Recently, the most successful approach in text classification is to use transformer neural networks (Vaswani et al., 2017), pretrained on large monolingual corpora, and then fine-tune them for a specific task, such as text classification. For example, BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2019) uses masked language modeling and order of sentences prediction tasks to build a general language understanding model. During the fine-tuning for a specific downstream task, additional layers are added to the BERT model, and the model is trained on the data of interest to capture the specific knowledge required to perform the task.

The main aims of the paper is to propose methods that can save time and resources during the text annotation process and improve prediction performance. As a test domain we use hate speech detection in tweets, news comments and Facebook comments. We investigate two performance improving techniques which can be summarized as our main contributions as follows.

1. We remove instances with uncertain classifications from the training set and show that fine-tuning on the cleaned dataset improves the performance of the BERT model. Less certain classifications can be selected for reannotation.

2. We combine predictions of machine learning models using the MM probabilistic ensemble method. The approach is beneficial for predictive performance.

The paper consists of five further sections. In Section 2, we present related works on prediction uncertainty and hate speech detection. In Section 3, we propose the methodology for uncertainty assessment of deep neural networks using attention layers and MCD. In Section 4, we describe the tested datasets and evaluation scenarios. The obtained results are presented in Section 5, followed by conclusions and ideas for further work in Section 6.

2 Related Work

In this section, we introduce related work split into four topics. First, we present the work on semi-supervised learning that can be used in text annotation, followed by the related research on Bayesian learning for text classification. In the third subsection, we describe probabilistic ensemble methods and in the fourth, we outline the related work on hate speech detection.

2.1 Semi-supervised Learning for Text Annotation

The performance of supervised learning depends on the availability of a sufficient amount of labeled data. However, manual labeling is expensive and difficult to scale up to large amounts of data. Semi-supervised learning tries to utilize large amounts of unlabeled data available for many problems by combining them with small amounts of labeled data (Zhu, 2005). The goal of semi-supervised learning is to understand how combining labeled and unlabeled data can change the learning behavior, and design algorithms that take advantage of such a combination (Zhu and Goldberg, 2009). Most semi-supervised learning strategies extend either unsupervised or supervised learning to include additional information typical of the other learning paradigm. The transductive learning is related to the semi-supervised learning, but assumes that the test set is known in advance and its goal is to optimize the generalization ability on this (unlabeled) test set (Zhou and Li, 2010). In the non-transductive setting, Acharya et al. (2013) combine probabilistic classifiers. They take class labels from existing classifiers and cluster labels from a clustering ensemble. The consensus labeling is assigned to the target data.

2.2 Bayesian Methods for Text Classification

Although, recent works on prediction uncertainty mostly investigate deep neural networks, many other probabilistic classifiers were analyzed in the past (Platt, 1999; Niculescu-Mizil and Caruana, 2005; Zhang

et al., 2013; Cao et al., 2015; He et al., 2018). Prediction reliability is an important issue for black-box models like neural networks as they do not provide interpretability or reliability information about their predictions. Most existing reliability scores for deep neural networks are contructed using Bayesian inference. The most popular exception is the work of Lakshminarayanan et al. (2017), who proposed to use deep ensembles to estimate the prediction uncertainty.

A computationally efficient simulation of Bayesian inference uses Monte Carlo dropout (Gal and Ghahramani, 2016a). The first implementation of dropout in recurrent neural networks (RNNs) was in 2013 (Wang and Manning, 2013) but further research revealed a negative impact of dropout in RNNs (Bluche et al., 2015). Later, the dropout was successfully applied to language modeling by Zaremba et al. (2014) who used it only in fully connected layers. Gal and Ghahramani (2016b) implemented the variational inference based dropout which can regularize also recurrent layers. In this way method mimics Bayesian inference by combining probabilistic parameter interpretation and deep RNNs. Several other works investigate how to estimate prediction uncertainty within different data frameworks using RNNs (Zhu and Laptev, 2017; Miok et al., 2019b), e.g., Bayes by Backpropagation (BBB) was applied to RNNs (Fortunato et al., 2017). Monte Carlo dropout was also introduced into variational autoencoders (Miok et al., 2019a; Miok et al., 2019c) and for estimating prediction intervals (Miok, 2018).

To our knowledge, Bayesian deep learning models were not yet used to detect less certain text classifications and remove them from a train dataset to improve the prediction performance.

2.3 Probabilistic Ensembles

Most methods used for text classification can produce probabilistic predictions which are rarely exploited beyond classification into a discrete class. As probabilistic predictions provide additional information compared to the discrete outcome, we use ensembles that can model predictive distributions. Ensemble methods can be divided into two main groups. The first group of methods estimates the performance of individual classifiers and weights them accordingly. The second group of methods learns the structure of predictions and bases their forecasts on it.

The first group of methods can be further divided into methods that are able to combine full posterior distributions and methods that only combine probabilistic point predictions. The advantage of the former is that they are more expressive, and a disadvantage is that they require inputs in the form of a full distribution, which is not always available. Bayesian model averaging (Hoeting et al., 1999) combines models by their marginal posterior probability. This method is suitable if one of the candidate models is the true data generating process, otherwise its performance decreases (Cerquides and De Mántaras, 2005). Bayesian stacking (Yao et al., 2018) is also useful in these cases. It is based on weighing the posterior predictive distributions of individual models by estimating their leave-one-out cross-validation performance. Linear opinion pool (Cooke, 1991) is a classical approach to combine classifiers and can be included into the second group of methods. It combines predictions as a linear combination of individual models by maximizing the likelihood. An example of Bayesian approach from the second group is the agnostic Bayesian learning of ensembles (Lacoste et al., 2014), which weighs the models by estimated probabilities of them being the best model; the estimates are based on holdout computation of generalization performance.

Methods that model the structure of predictions are especially useful in case of complex relationships between individual models' predictions and the response variable. Independent Bayesian classifier combination (IBCC) (Kim and Ghahramani, 2012) combines non-probabilistic predictions by estimating the probability mass of predictions with a categorical distribution, conditional on the true label. It provides probabilities for a new observation that are proportional to the probability mass of new inputs for each true label. Nazábal et al. (2016) has extended the IBCC to probabilistic predictions by using the Dirichlet distribution. Supra-Bayesian methods (Lindley, 1985) combine probabilistic predictions using the log-odds of probabilities and modeling them with the multivariate normal (MVN) distribution, conditional on the true label. They use the common covariance matrix over all true labels but vary the means.

Ensemble modeling has recently been studied also within the text classification area (Li et al., 2018; Silva et al., 2010; Kilimci and Akyokus, 2018), but not within the context of probabilistic ensemble

models. In our work, we investigate Bayesian ensemble modeling for hate speech detection and how this can improve individual model predictions.

2.4 Hate Speech Detection

Analyzing sentiments and extracting emotions from the text are useful natural language processing applications (Sun et al., 2018). Being one of the wide range of applications where machines tend to understand human sentiments, hate speech detection is gaining importance with the rise of social media. We regard hate speech as written or oral communication that abuses or threatens a specific group or target (Warner and Hirschberg, 2012).

Detecting abusive language for less-resourced languages is difficult, hence, multilingual and cross-lingual methods are employed to improve the results (Stappen et al., 2020). This is especially the case when the involved languages are morphologically or geographically similar (Pamungkas and Patti, 2019). In our work, we investigate and compare hate speech detection methods for English, Croatian, and Slovene. English is by far the most researched language with plenty of resources (Malmasi and Zampieri, 2017; Davidson et al., 2017; Waseem and Hovy, 2016). Recently, hate speech detection studies were done also on neighbouring Slavic languages Croatian (Kocijan et al., 2019; Ljubešić et al., 2018) and Slovene (Fišer et al., 2017; Ljubešić et al., 2019; Vezjak, 2018).

Hate speech detection is usually treated as a binary text classification problem, and is approached with supervised learning methods. In the past, the most frequently used classifier was the Support Vector Machine (SVM) method (Schmidt and Wiegand, 2017), but recently deep neural networks showed superior performance, first through recurrent neural networks (Mehdad and Tetreault, 2016), and recently using large pretrained transformer networks (Mozafari et al., 2019; Wiedemann et al., 2020). In this work, we use the recent state-of-the-art pretrained (multilingual) BERT model.

3 Methods

We describe two approaches to the assessment of prediction reliability, Bayesian Attention Networks and Bayesian Probabilistic Ensembles.

3.1 Bayesian Attention Networks

The work (Miok et al., 2020) that introduce method named 'Bayesian Attention Networks' (BAN), proposes the dropout layers to be active also during the prediction phase. In this way, predictions are rather random and are sampled from the *learned* distribution, thereby forming an ensemble of predictions. The obtained distribution can be, for example, inspected for higher moment properties and it can offer additional information on the certainty of a given prediction. During the prediction phase, all layers of the network except the dropout layers are deactivated. The forward pass on such partially activated architecture is repeated for a fixed number of samples, each time producing a different outcome that can be combined into the final probability, or inspected as a probability distribution.

Monte Carlo dropout was adapted for the BERT model in the same way as for BANs. MCD can provide multiple predictions during the test time without any additional training (Gal, 2016). Training a neural network with dropout spreads the information contained in the neurons across the network. Hence, during the prediction, such a trained neural network will be robust; using the dropout principle, a new prediction is created in each forward pass, and a sufficiently large set of such predictions can be used to estimate prediction reliability. The BERT models are trained with 10% of dropout in all of the layers by default. Therefore, it allows for multiple predictions with the fine-tuned model. We call this model MCD BERT. A possible limitation of this approach is that during training a single dropout rate of 10% is used, while other dropout probabilities might be more suitable for reliability estimation. We leave this question for further work as it requires long and costly training of several BERT models.

3.2 Bayesian Probabilistic Ensemble

To alleviate the drawbacks of individual classification models, we propose the use of MM (Pirš and Štrumbelj, 2019), a Bayesian ensemble method suitable for combining correlated probabilistic predictions.

MM is an extension of IBCC (Kim and Ghahramani, 2012), which combines non-probabilistic predictions. The method is based on finding the latent structure of combined predictions and provides new probabilities based on its distribution. Let m be the number of classes and r the number of individual models we are combining. The main idea is similar to Supra-Bayesian ensembles (Lindley, 1985), as we first transform individual probabilistic predictions with the inverse logistic transformation (log-odds) to move from [0,1] space to the $\mathbb{R}$ space. We merge the transformed predictions of individual models and get a $(m-1)r$-variate distribution. We model this latent distribution with multivariate normal mixtures, conditional on the true label in a similar fashion as in the case of linear discriminant analysis. Let θ represent estimated parameters and θ_t the subset of parameters estimated for observations with true label t. Let $T^* \in \{1, 2, ..., m\}$ be the response random variable for a new observation and $u^* \in \mathbb{R}^{(m-1)r}$ the transformed and merged predictions for this new observation. Probabilistic predictions for unseen data can then be generated by calculating the densities of merged predictions for new data:

$$p(T^* = t|u^*, \theta) = \frac{p(u^*|\theta_t)(\gamma_t n_t)}{\sum_{i=1}^{r} p(u^*|\theta_i)(\gamma_i n_i)},$$

where p is the MVN mixture probability density, γ_t is the frequency prior for class t, and n_t is the number of true labels in class t in the training dataset. The method uses a regularization term, which increases the variance in any dimension that is difficult to model or has a detrimental effect on the results, effectively decreasing its effect. For a complete Bayesian specification and the derivation of the Gibbs sampler, we refer the reader to (Pirš and Štrumbelj, 2019). We used the same priors as proposed in this paper.

MM is well-suited for combining biased classifiers, or classifiers with systematic errors. It can serve as a calibration tool for an individual classifier by learning its latent distribution. Since BERT is usually accurate but less well calibrated, the MM method has the potential to alleviate miscalibration, while improving or at least preserving the classification performance.

4 Experimental Setting

We first introduce the three phases of our experiments, followed by the used datasets and implementation details. The experimental setting consists of three phases:

1. We categorize classifications to trusted and untrusted based on the uncertainty measure from MCD BERT. In this way, we can detect borderline classification that make a false impression of certainty.

2. We remove the instances with uncertain classifications from the *training set* to improve the dataset on which the BERT model is fine-tuned. This provides better quality data for training and shall improve the quality of the resulting prediction model.

3. We use Bayesian ensemble to combine automatic predictions with annotators' decisions to remove low-quality training instances.

4.1 Datasets

To test the proposed methodology in the multilingual context, we trained the presented classification models on three different datasets, summarized in Table 1.

1. The **English** dataset[1] is extracted from the hate speech and offensive language detection study of Davidson et al. (2017). We used the subset of data consisting of 5,000 tweets. We took 1,430 tweets labeled as hate speech and randomly sampled 3,670 tweets from the collection of the remaining 23,353 tweets.

2. The **Croatian** dataset was provided by the Styria media company within the EU Horizon 2020 EMBEDDIA project[2]. The texts were extracted from user comments in the news portal Večernji list[3].

[1] https://github.com/t-davidson/hate-speech-and-offensive-language
[2] http://embeddia.eu
[3] https://www.vecernji.hr

The original dataset consists of 9,646,634 comments from which we selected 8,422 comments of which 50% are labeled as hate speech by human moderators and the other half was randomly chosen from the non-problematic comments.

3. **Slovene** dataset is a result of the Slovenian national project FRENK[4]. Our dataset comes from two studies on Facebook comments (Ljubešić et al., 2019). The first study deals with LGBT homophobia topics while the second analyzes anti-migrants posts. We used all 2,188 hate speech comments, and randomly sampled 3,812 non-hate speech comments.

Table 1: Characteristics of the used datasets: type and number of instances, as well as the input embeddings for each of the datasets.

Dataset	type	Size	Hate	Non-hate	LSTM embeddings
English	tweets	5000	1430	3670	sentence
Croatian	news comments	8422	4211	4211	fastText
Slovene	Facebook comments	6000	2188	3812	fastText

4.2 Implementation

The two Bayesian methods that were proposed in this paper to improve the annotation process have full implementation within their original papers. All of the particularities of how MCD BERT was implemented in PyTorch library[5] are presented in (Miok et al., 2020). The implementation details of the MM method are clearly explained in (Pirš and Štrumbelj, 2019) methods section and in this paper we provide full R code [6].

5 Results

In this section, we present three groups of results: removing uncertain instances from the training set, creating a cleaner training set, and improving annotations using he Bayesian ensembles.

5.1 Removing Uncertain Instances

Using MCD BERT, we obtain multiple predictions for each test set instance, and compute their mean and variance. Using the mean, we determine the classification (hate speech or not), while the variance reports on the certainty of the BERT for this specific instance. Based on the variance, we group classifications into certain and uncertain. Unsurprisingly, removing the uncertain test set instances improves the prediction performance as shown in Table 2, but also leaves a portion of borderline instances unclassified.

From Table 2 we can conclude that the variance of MCD BERT predictions is correlated with the performance of models: the more variance there is in the predictions the less accurate the model. Thus, removing the uncertain classifications can seemingly improve the performance of the test set. A practical benefit of this is that uncertain classification could be passed back to annotators to recheck them.

5.2 Creating Cleaner Training Sets

While the removal of uncertain instances from the test set might just sweep the problematic instances under the carpet, a more practical benefit is to use the uncertainty information to create a better training set. The test tweets/comments were removed based on how variate are their predictions. Thus, we repeatedly train the MCD BERT model on part of the dataset and use this model to obtain multiple predictions on the other part of the training dataset. In such a way, we collect multiple predictions for all original training tweets or comments and remove observations with the highest prediction variance. As a result of this procedure, 15 and 18 percent of the most uncertain predictions were removed for the English and Slovene dataset respectively. Croatian dataset contains a lot of comments with high variability in their predictions

[4] `http://nl.ijs.si/frenk/` (Research on Electronic Inappropriate Communication)
[5] `https://github.com/KristianMiok/Bayesian-BERT`
[6] `https://github.com/gregorp90/MM`

Table 2: Performance of multilingual BERT model, after removing uncertain instances from the test set of 1000 comments.

Language	Metric	Full dataset	200 removed	500 removed	700 removed
EN	**Accuracy**	0.91	0.96	0.996	0.997
	Precision	0.90	0.95	0.992	0.994
	Recall	0.89	0.95	1	1
	F1	0.88	0.95	0.995	0.997
CRO	**Accuracy**	0.72	0.76	0.84	0.87
	Precision	0.68	0.71	0.80	0.85
	Recall	0.54	0.69	0.78	0.75
	F1	0.61	0.70	0.79	0.83
SLO	**Accuracy**	0.71	0.76	0.83	0.87
	Precision	0.60	0.65	0.70	0.65
	Recall	0.56	0.64	0.66	0.54
	F1	0.58	0.65	0.68	0.59

so for this dataset we removed around 35% of the most uncertain comments. The details of how many instances were removed for each of the three datasets are presented in Table 3.

Table 3: Sizes of the datasets before and after the removing: original number of instances, number of instances removed and final training data size.

Dataset	Training Size	Number of removed	Final Size	Percent removed
English	4000	719	3281	18 %
Croatian	7422	2615	4807	35%
Slovene	5000	731	4269	15 %

Using prediction certainty to remove the uncertain instances from the training can improve the fine-tuning of BERT. For neural network models, during training or fine-tuning their performance is evaluated on a separate validation set. In Table 4, we can observe how the prediction accuracy on the validation set is improved with number of training epochs. We can see that fine-tuning BERT on the cleaner dataset improves its performance. We hypothesize that when the uncertainty due to unreliable labels is reduced, the decision boundary is easier to determine.

Table 4: Performance (measured using F_1 score) on the validation sets during training for original and cleaned datasets.

	English		Croatian		Slovene	
	Original	Cleaned	Original	Cleaned	Original	Cleaned
Epoch1	0.92	0.98	0.68	0.77	0.70	0.64
Epoch2	0.92	0.98	0.69	0.77	0.70	0.77
Epoch3	0.92	0.98	0.68	0.78	0.71	0.79
Epoch4	0.92	0.98	0.70	0.79	0.72	0.81

Results for the model fine-tuned on the cleaned dataset are contained in Table 5. Compared to the results in Table 2 (see the "Full dataset" column), the prediction results for Croatian and Slovenian datasets are improved while for the English dataset this is not the case. We explain this by the fact that the English dataset is well-annotated with high-quality predictions. On the other hand, we believe that the Croatian and Slovenian datasets are less clean and contain several questionable annotations. This can be confirmed for the Croatian dataset, which was created within the project we participate in, so we are well-informed about the annotation process.

Table 5: Test set performance (F_1 score) of the models trained on the cleaned datasets.

Metrics	English	Croatian	Slovene
Accuracy	0.87	0.74	0.72
Precision	0.88	0.73	0.62
Recall	0.81	0.60	0.55
F1	0.85	0.66	0.59

5.3 Improving Annotations using Bayesian Ensembles

We propose a Bayesian ensemble as a support method for the annotation process. As annotators can be distracted, biased, or influenced, we propose to use the MM method to provide them a hint of how shall they annotate the instances. From Table 6, we can observe that by combining probabilistic predictions of BERT, random forest, and support vector machines, we can further improve the predictive performance. The MM ensemble not only improves BERT's results but also provides better calibrated predictions as evidenced from Figure 1.

Table 6: The F_1 score of the hate speech classifiers and their ensemble.

Method	English	Croatian	Slovene
BERT	0.91	0.72	0.71
RF	0.83	0.67	0.65
SVM	0.86	0.71	0.69
MM	**0.92**	**0.74**	**0.72**

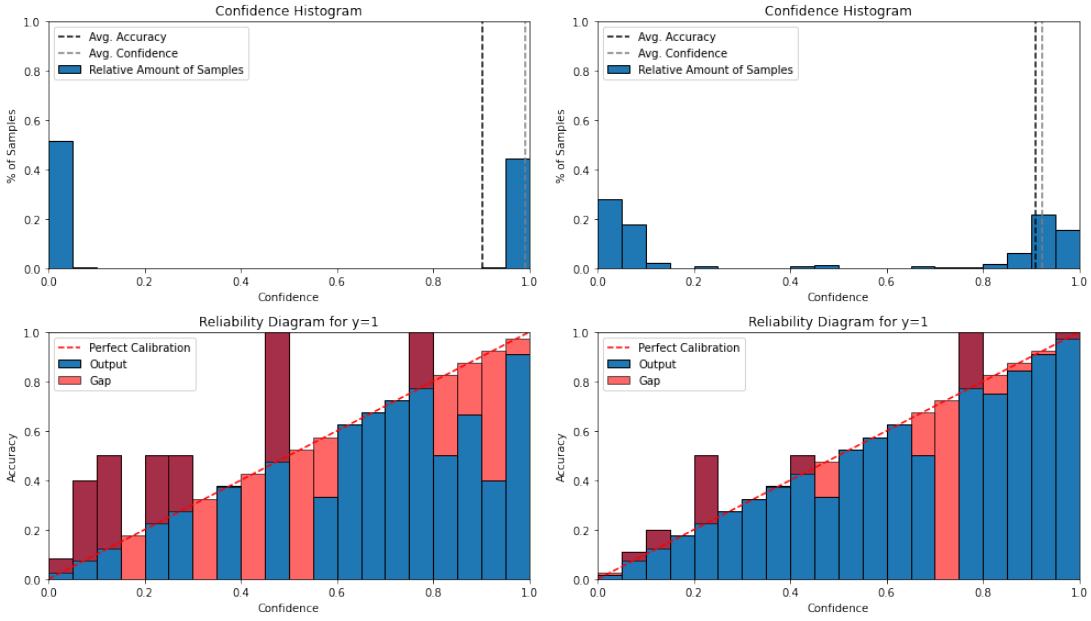


Figure 1: Calibration for the BERT predictions (left) and MM model predictions (right).

6 Conclusions and Further Work

A large amount of currently available textual data allows and requires modeling with machine learning methods. To apply the supervised methods, the text data has to be annotated, and effective learning requires accurate annotations, which may be expensive for organizers and difficult for human annotators. For this reason, the annotation process is often coupled with semi-supervised machine learning and classification reliability estimation.

We presented several machine learning approaches, based on Bayesian inference, that can improve the data annotation process. First, multiple predictions obtained with MCD BERT can identify instances with questionable labeling. Second, removing training instances with unreliable labels can improve the quality of the training set, making it more homogeneous and cleaner, thereby improving the predictive performance of BERT models. Third, probabilistic ensemble combinations can help annotators to better label the data by providing more accurate and better calibrated prediction probabilities. In conclusion, Bayesian methods can improve the annotation process and shall be further investigated and improved for this task.

In further work, we will focus on improving our method on how to remove uncertain instances. We will construct and test a workflow for semi-supervised text annotation in a real-world setting. Testing different dropout levels in the BERT model may provide a better understanding of its uncertainty and calibration.

Acknowledgements

This paper was supported by European Union's Horizon 2020 Programme project EMBEDDIA (Cross-Lingual Embeddings for Less-Represented Languages in European News Media, grant no. 825153). The research was supported by the Slovenian Research Agency through research core funding no. P6-0411, project CANDAS (Computer-assisted multilingual news discourse analysis with contextual embeddings, grant no. J6-2581), and Young researcher grant (Gregor Pirš).

References

Ayan Acharya, Eduardo R Hruschka, Joydeep Ghosh, Badrul Sarwar, and Jean-David Ruvini. 2013. Probabilistic combination of classifier and cluster ensembles for non-transductive learning. In *Proceedings of the 2013 SIAM International Conference on Data Mining*, pages 288–296. SIAM.

Théodore Bluche, Christopher Kermorvant, and Jérôme Louradour. 2015. Where to apply dropout in recurrent neural networks for handwriting recognition? In *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pages 681–685. IEEE.

Keyan Cao, Guoren Wang, Donghong Han, Jingwei Ning, and Xin Zhang. 2015. Classification of uncertain data streams based on extreme learning machine. *Cognitive Computation*, 7(1):150–160.

Jesús Cerquides and Ramon López De Mántaras. 2005. Robust Bayesian linear classifier ensembles. In *European Conference on Machine Learning*, pages 72–83. Springer.

Roger Cooke. 1991. *Experts in uncertainty: opinion and subjective probability in science*. Oxford University Press on Demand.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In *Eleventh international AAAI conference on web and social media.*

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Darja Fišer, Tomaž Erjavec, and Nikola Ljubešić. 2017. Legal framework, dataset and annotation schema for socially unacceptable online discourse practices in slovene. In *Proceedings of the first workshop on abusive language online*, pages 46–51.

Meire Fortunato, Charles Blundell, and Oriol Vinyals. 2017. Bayesian recurrent neural networks. *arXiv preprint arXiv:1704.02798*.

Yarin Gal and Zoubin Ghahramani. 2016a. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, pages 1050–1059.

Yarin Gal and Zoubin Ghahramani. 2016b. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1019–1027.

Yarin Gal. 2016. Uncertainty in deep learning. *University of Cambridge*, 1:3.

Lirong He, Bin Liu, Guangxi Li, Yongpan Sheng, Yafang Wang, and Zenglin Xu. 2018. Knowledge base completion by variational bayesian neural tensor decomposition. *Cognitive Computation*, 10(6):1075–1084.

Jennifer A. Hoeting, David Madigan, Adrian E. Raftery, and Chris T. Volinsky. 1999. Bayesian model averaging: a tutorial. *Statistical science*, pages 382–401.

Pavel Izmailov, Wesley J Maddox, Polina Kirichenko, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. 2019. Subspace inference for bayesian deep learning. *arXiv preprint arXiv:1907.07504*.

Zeynep H Kilimci and Selim Akyokus. 2018. Deep learning-and word embedding-based heterogeneous classifier ensembles for text classification. *Complexity*, 2018.

Hyun-Chul Kim and Zoubin Ghahramani. 2012. Bayesian Classifier Combination. In *International Conference on Artificial Intelligence and Statistics*, pages 619–627.

Kristina Kocijan, Lucija Košković, and Petra Bajac. 2019. Detecting hate speech online: A case of croatian. In *International Conference on Automatic Processing of Natural-Language Electronic Texts with NooJ*, pages 185–197. Springer.

Alexandre Lacoste, Mario Marchand, François Laviolette, and Hugo Larochelle. 2014. Agnostic Bayesian learning of ensembles. In *International Conference on Machine Learning*, pages 611–619.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. 2017. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in neural information processing systems*, pages 6402–6413.

Ming Li, Peilun Xiao, and Ju Zhang. 2018. Text classification based on ensemble extreme learning machine. *arXiv preprint arXiv:1805.06525*.

Dennis V. Lindley. 1985. Reconciliation of discrete probability distributions' in Bernado. *JM, DeGroot, MH, Lindley, DV, and Smith, AFM, Eds., Bayesian Statistics II. North Holland, Amsterdam*, pages 375–391.

Nikola Ljubešić, Tomaž Erjavec, and Darja Fišer. 2018. Datasets of slovene and croatian moderated news comments. In *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*, pages 124–131.

Nikola Ljubešić, Darja Fišer, and Tomaž Erjavec. 2019. The FRENK datasets of socially unacceptable discourse in Slovene and English. In *International Conference on Text, Speech, and Dialogue*, pages 103–114. Springer.

Shervin Malmasi and Marcos Zampieri. 2017. Detecting hate speech in social media. *arXiv preprint arXiv:1712.06427*.

Yashar Mehdad and Joel Tetreault. 2016. Do characters abuse more than words? In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 299–303.

Kristian Miok, Dong Nguyen-Doan, Marko Robnik-Šikonja, and Daniela Zaharie. 2019a. Multiple imputation for biomedicaldata using monte carlo dropout autoencoders. In *7th IEEE International Conference on E-Health and Bioengeneering (EHB)*. IEEE.

Kristian Miok, Dong Nguyen-Doan, Blaž Škrlj, Daniela Zaharie, and Marko Robnik-Šikonja. 2019b. Prediction uncertainty estimation for hate speech classification. In *International Conference on Statistical Language and Speech Processing*, pages 286–298. Springer.

Kristian Miok, Dong Nguyen-Doan, Daniela Zaharie, and Marko Robnik-Šikonja. 2019c. Generating data using monte carlo dropout. In *2019 IEEE 15th International Conference on Intelligent Computer Communication and Processing (ICCP)*, pages 509–515. IEEE.

Kristian Miok, Blaz Skrlj, Daniela Zaharie, and Marko Robnik-Sikonja. 2020. To ban or not to ban: Bayesian attention networks for reliable hate speech detection. *arXiv preprint arXiv:2007.05304*.

Kristian Miok. 2018. Estimation of prediction intervals in neural network-based regression models. In *2018 20th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, pages 463–468. IEEE.

Marzieh Mozafari, Reza Farahbakhsh, and Noel Crespi. 2019. A BERT-based transfer learning approach for hate speech detection in online social media. In *International Conference on Complex Networks and Their Applications*, pages 928–940. Springer.

Pavel Myshkov and Simon Julier. 2016. Posterior distribution analysis for bayesian inference in neural networks. *Advances in Neural Information Processing Systems (NIPS)*.

Alfredo Nazábal, Pablo García-Moreno, Antonio Artés-Rodríguez, and Zoubin Ghahramani. 2016. Human activity recognition by combining a small number of classifiers. *IEEE journal of biomedical and health informatics*, 20(5):1342–1351.

Alexandru Niculescu-Mizil and Rich Caruana. 2005. Predicting good probabilities with supervised learning. In Luc De Raedt and Stefan Wrobel, editors, *Proceedings of the 22nd International Machine Learning Conference*. ACM Press.

Endang Wahyu Pamungkas and Viviana Patti. 2019. Cross-domain and cross-lingual abusive language detection: A hybrid approach with deep learning and a multilingual lexicon. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 363–370.

Gregor Pirš and Erik Štrumbelj. 2019. Bayesian combination of probabilistic classifiers using multivariate normal mixtures. *J. Mach. Learn. Res.*, 20:51–1.

John C. Platt. 1999. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in large margin classifiers*, pages 61–74. MIT Press.

Anna Schmidt and Michael Wiegand. 2017. A survey on hate speech detection using natural language processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*, pages 1–10.

Catarina Silva, Uros Lotric, Bernardete Ribeiro, and Andrej Dobnikar. 2010. Distributed text classification with an ensemble kernel-based learning approach. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 40(3):287–297.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Lukas Stappen, Fabian Brunn, and Björn Schuller. 2020. Cross-lingual zero-and few-shot hate speech detection utilising frozen transformer language models and axel. *arXiv preprint arXiv:2004.13850*.

Xiao Sun, Xiaoqi Peng, and Shuai Ding. 2018. Emotional human-machine conversation generation based on long short-term memory. *Cognitive Computation*, 10(3):389–397.

György Szarvas, Veronika Vincze, Richárd Farkas, and János Csirik. 2008. The bioscope corpus: annotation for negation, uncertainty and their scope in biomedical texts. In *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing*, pages 38–45.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Boris Vezjak. 2018. Radical hate speech: The fascination with Hitler and fascism on the Slovenian webosphere. *Solsko Polje*, 29.

Veronika Vincze. 2015. *Uncertainty detection in natural language texts*. Ph.D. thesis, szte.

Sida Wang and Christopher Manning. 2013. Fast dropout training. In *International Conference on Machine Learning*, pages 118–126.

William Warner and Julia Hirschberg. 2012. Detecting hate speech on the world wide web. In *Proceedings of the second workshop on language in social media*, pages 19–26. Association for Computational Linguistics.

Zeerak Waseem and Dirk Hovy. 2016. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In *Proceedings of the NAACL student research workshop*, pages 88–93.

Gregor Wiedemann, Seid Muhie Yimam, and Chris Biemann. 2020. Uhh-lt & lt2 at semeval-2020 task 12: Fine-tuning of pre-trained transformer networks for offensive language detection. *arXiv preprint arXiv:2004.11493*.

Yuling Yao, Aki Vehtari, Daniel Simpson, Andrew Gelman, et al. 2018. Using stacking to average Bayesian predictive distributions. *Bayesian Analysis*, 13(3):917–1007.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

Xunan Zhang, Shiji Song, and Cheng Wu. 2013. Robust bayesian classification with incomplete data. *Cognitive Computation*, 5(2):170–187.

Zhi-Hua Zhou and Ming Li. 2010. Semi-supervised learning by disagreement. *Knowledge and Information Systems*, 24(3):415–439.

Zhi-Hua Zhou. 2012. *Ensemble methods: foundations and algorithms*. CRC press.

Xiaojin Zhu and Andrew B Goldberg. 2009. Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning*, 3(1):1–130.

Lingxue Zhu and Nikolay Laptev. 2017. Deep and confident prediction for time series at Uber. In *IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 103–110.

Xiaojin Jerry Zhu. 2005. Semi-supervised learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences.

Appendix D: Bayesian BERT for Trustful Hate Speech Detection

Bayesian BERT for Trustful Hate Speech Detection

Kristian Miok [1 2] **Blaž Škrlj** [3] **Daniela Zaharie** [1] **Marko Robnik Šikonja** [2]

Abstract

Hate speech is an important problem in the management of user-generated content. In order to remove offensive content or ban misbehaving users, content moderators need reliable hate speech detectors. Recently, deep neural networks based on transformer architecture, such as (multilingual) BERT model, achieve superior performance in many natural language classification tasks, including hate speech detection. So far, these methods have not been able to quantify their output in terms of reliability. We propose a Bayesian method using Monte Carlo Dropout within the attention layers of the transformer models to provide well-calibrated reliability estimates. We evaluate the introduced approach on hate speech detection problems in several languages. Our approach not only improves the classification performance of the state-of-the-art multilingual BERT model but the computed reliability scores also significantly reduce the workload in inspection of offending cases and in reannotation campaigns.

1. Introduction

With the rise of the social network popularity, hate speech phenomena has significantly increased (Davidson et al., 2017). Hate speech not only harms both minority groups and the whole society but it can lead to actual crimes (Bleich, 2011). Hence, automated hate speech detection mechanisms are urgently needed. On the other hand, falsely accusing people of hate speech is also a problem. Many content providers rely on human moderators to reliably decide if a given context is offensive or not but this is a mundane and stressful job which can even cause post-traumatic stress disorders[1]. There are many attempts to automate detection of hate speech in the social media using machine learning, but existing models lack quantification of reliability for their decisions. In the last few years, recurrent neural networks (RNNs) were the most popular choice in text classification. The LSTM networks, the most successful RNN architecture, were already successfully adapted for assessment of predictive reliability in hate speech classification (Miok et al., 2019b). Recently, neural network architecture with attention layers, called transformer architecture (Vaswani et al., 2017), shows even better performance on almost all language processing tasks. Using transformer networks for the task of masked language modelling produced breakthrough pretrained models such as BERT (Devlin et al., 2018). Hence, the attention mechanism seems to be at the forefront of natural language understanding with potentially huge impact on language applications. We aim to investigate the behavior of the attention mechanism concerning the reliability of predictions. We focus on the hate speech recognition task.

In hate speech detection, reliable predictions are needed to remove harmful contents and possibly ban malicious users without harming the freedom of speech (Miok et al., 2019b). Standard neural networks are inadequate for assessment of predictive uncertainty, and the Bayesian framework is the principled approach to doing so. However, classical Bayesian inference techniques do not scale well in neural networks with high dimensional parameter space (Izmailov et al., 2019). Various methods were proposed in order to overcome this problem (Myshkov & Julier, 2016). One of the most efficient method is called Monte Carlo Dropout (MCD) (Gal & Ghahramani, 2016). Its idea is to use dropout in neural networks as a regularization technique (Srivastava et al., 2014) and interpret it as a method that mimics Bayesian approach.

We propose a model that combines the attention mechanism in transformer networks with the MCD based Bayesian inference in order to estimate reliability of hate speech predictions. Our main contributions are estimating prediction uncertainty of the attention network (AN) and BERT model and testing the proposed reliability methods within the multilingual hate speech detection tasks.

*Equal contribution [1]Department of Computer Science, West University of Timisoara, Timisoara, Romania [2]Faculty of Computer and Information Science, University of Ljubljana, Ljubljana, Slovenia [3]Jožef Stefan International Postgraduate School, Jožef Stefan Institute, Ljubljana, Slovenia. Correspondence to: Kristian Miok <kristian.miok@e-uvt.ro>.

Presented at the ICML 2020 Workshop on Uncertainty and Robustness in Deep Learning.

[1] https://www.bbc.com/news/technology-51245616

The paper consists of four more sections. In Section 2, we propose the methodology for uncertainty assessment using attention layers and MCD. Section 3 presents the data sets and the evaluation scenario. The obtained results are presented in Section 4, followed by the conclusions in Section 5.

2. Bayesian Attention Networks

The BERT model (Devlin et al., 2018) is the representative of transformer networks and has achieved state-of-the art results in many NLP tasks, including text classification (Xu et al., 2020; Gururangan et al., 2019; Chang et al., 2019). In this work, we introduce Monte Carlo Dropout to transformer networks and BERT with the intention to construct their Bayesian variants. Analysis of different amounts of dropout, different BERT variants modifications, and their hyperparameters would require huge computational resources, e.g., training a single BERT model on four TPUs requires more than a month time. Due to limited computational resources, we explore these issues in a limited setting, first on only the encoder part of the BERT architecture, called Attention Network (AN), and then on the entire pretrained BERT model.

In the following subsections, we first formally define the Attention Network architecture, and then make it Bayesian by introducing MCD. We describe how we can introduce MCD principle into the already pretrained BERT model.

2.1. Attention Networks

The basic architecture of Attention Network follows the architecture of transformer networks (Vaswani et al., 2017) and is shown in Figure 1. The architecture is similar to the

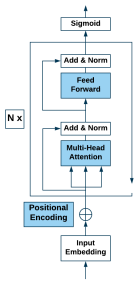


Figure 1. A scheme of Attention Networks. In layers colored blue we introduce the dropout.

encoder part of the transformer architecture. The difference is in the output part where a single output head was added to perform either binary classification, using the sigmoid activation function. By applying only the encoder part of transformer architecture, orders of magnitude less parameters are needed to learn a particular classification task, e.g., in this work, we used at maximum 3 million parameters. The architecture can contain many attention heads, where a single attention heads is computed as:

$$o_h = \text{softmax}(\frac{\boldsymbol{Q} \cdot \boldsymbol{K}^T}{\sqrt{d_k}}) \cdot \boldsymbol{V},$$

The attention matrices are commonly known as the query $\boldsymbol{Q}$, the key $\boldsymbol{K}$, and the value matrix $\boldsymbol{V}$. The o_h represents the output. The attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key. The d_k represents the dimensionality of the keys. The positional encoding, as discussed in (Vaswani et al., 2017), represents a matrix that encodes individual positions in a matrix of same dimensionality as the one holding the information on sequences (input embedding).

2.2. Monte Carlo Dropout for Attention Networks

MCD was recently used within various models and architectures in order to obtain the prediction uncertainty and improve the classification results (Miok, 2018; Miok et al., 2019c;a). Transformer networks were not analyzed yet. In our proposal, called Monte Carlo Dropout Bayesian Attention Networks (BAN or MCD AN) contrary to the original dropout setting, the dropout layers are active also during the prediction phase. In this way, predictions are not constant and are sampled from the *learned* distribution, thereby forming an ensemble of predictions. The obtained distribution can be, for example, inspected for higher moment properties and can offer additional information on the certainty of a given prediction. During the prediction phase, all layers except the dropout layers are deactivated. Forward pass on such partially activated architecture is repeated for a fixed number of samples, which can be combined to obtain the final probability, or further inspected as a distribution underlying the probability.

2.3. Monte Carlo Dropout for BERT

Monte Carlo dropout was used for the BERT predictions in the same way as for BAN. MCD can provide multiple predictions during the test time completely free, as long as the dropout was used during the training time (Gal, 2016). Training neural network with dropout distributes the information contained in the neurons throughout the network. Hence, during the prediction, such trained neural network will be robust; using the dropout principle a new prediction is possible in each forward pass, and sufficiently large set of such predictions can be used to estimate the reliability. BERT model is trained with 10% of dropout in all of the

layers and thus allows for multiple predictions using the described principle. We call this model MCD BERT and it natuarally provides reliability estimates. A possible limitation of this approach is that during training a single dropout rate of 10% is used, while other dropout probabilities might be more suitable for reliability estimation.

3. Evaluation Setting

We evaluate the proposed novelties concerning two main aspects: the calibration of returned probabilities, prediction performance and prediction uncertainty estimation. We first describe the hate speech data sets used, followed by the implementation details. In the last two subsections, we present evaluation measures for prediction performance and calibration.

3.1. Hate Speech Datasets

In order to test the proposed methodology in the multilingual context, we applied our models to three different data sets.

1. **English** data set[2] originates in a study regarding hate speech detection and the problem of offensive language (Davidson et al., 2017). Our data set consists of 5000 tweets. We took 1430 tweets labeled as hate speech and randomly sampled 3670 tweets from the collection of remaining 23353 tweets.

2. **Croatian** data set was collected by the Styria company within the EU Horizon 2020 project EMBEDDIA[3]. The text was extracted from the database of user comments, from the vecernji.hr[4] news portal. The original data set consists 9,646,634 comments described with 11 attributes from which we selected 8422 comments, one half of which were labelled as hate speech by human moderators, the other half was randomly chosen from the non-problematic comments.

3. **Slovenian** data set is a result of the Slovene national project FRENK[5]. The text data set used in the experiment was the combination of two different studies made on Facebook comments on the LGBT homophobia and anti-migrants published in the (Ljubešić et al., 2019). For the final data set we select 2182 hate and 2182 non-hate speech comments.

[2] https://github.com/t-davidson/hate-speech-and-offensive-language

[3] http://embeddia.eu

[4] https://www.vecernji.hr

[5] "FRENK - Raziskave Elektronske Nespodobne Komunikacije" (engl. "Research on Electronic Inappropriate Communication")

3.2. Prediction models

We used three types of prediction models: MCD LSTM networks (Miok et al., 2019b), MCD Bayesian Attention Networks (MCD AN) and MCD BERT. As the input to MCD LSTM we used pretrained word embeddings, sentence encoder for English (Cer et al., 2018) and fastText[6] for Slovenian and Croatian. For MCD AN we used simple tokenizer[7]. For the MCD BERT we used BERT's tokeizer. The summary is collected in Table 2.

3.3. Implementation details

We implemented the proposed MCD ANs in PyTorch[8]. The main hyperparameters of the architecture are the number of attention heads and the number of attention layers. The proposed adaptive classification threshold is computed after each validation set evaluation, i.e. every time we compute the performance on the validation set.'

Other parameters are set as follows. We use the Adamax optimizer (Kingma & Ba, 2014), a variant of Adam based on infinity norm. Binary cross-entropy loss guides the training. In order to automatically stop training, we use the stopping step of 10 – if after 10 optimization steps the performance on the validation set is not improved, the training stops.

We explored the following hyperparameter tuning space: the validation percentage (size of validation set) was varied between 5% and 10%. The rationale for testing different percentages of validation set sizes is that the data considered is small, hence considering too high validation percentages could omit the classifier from viewing crucial instances and thus reduce its final performance. Given enough data, however, the percentage should be as high as possible. Number of epochs was either 30 or 100, number of hidden layers and attention heads was 1 or 2. Maximum padding of the input sequences was either 48, 32 or 64. Learning rate was either 0.001 or 0.0005 and the adaptive threshold was either enabled or disabled.

MCD LSTM networks consist of an embedding layer, LSTM layer, and a fully connected layer within the Word2Vec and ELMo embeddings. In order to obtain best architectures for the LSTM and MCD LSTM models, various number of units, batch size, dropout rates and so on were fine-tuned. For BERT implementation the BERT base was implemented for both English and multilingual versions using Hugging Face code [9].

[6] https://fasttext.cc

[7] https://keras.io/preprocessing/text/

[8] https://gitlab.com/skblaz/bayesianattention

[9] https://huggingface.co/transformers/model_doc/bert.html

Bayesian BERT for Trustful Hate Speech Detection

Table 1. Comparison of predictive models using sentence embeddings. We present average classification accuracy, precision, recall and F_1 score (and standard deviations), computed using 5-fold cross-validation. All the results are expressed in percentages and the best results for each language is typeset in bold.

	English Tweets				Croatian Comments				Slovenian Comments			
Model	**Accuracy**	**Precision**	**Recall**	**F1**	**Accuracy**	**Precision**	**Recall**	**F1**	**Accuracy**	**Precision**	**Recall**	**F1**
MCD LSTM	81.0 [1.2]	81.5 [1.8]	82.5 [2.7]	81.9 [1.3]	63.7 [1.0]	68.5 [1.2]	40.8 [4.0]	51.0 [3.3]	55.3 [0.69]	53.5 [4.27]	57.0 [9.55]	43.13 [0.8]
MCD AN	83.3 [1.7]	80.5 [3.47]	82.8 [3.9]	81.6 [3.4]	61.4 [2.0]	58.6 [9.3]	30.2 [11.0]	38.1 [8.6]	57.4 [1.7]	49.3 [3.7]	27.9 [7.8]	35.1 [6.3]
BERT	90.9 [0.7]	89.3 [1.4]	90.8 [1.3]	90.0 [0.7]	70.8 [1.0]	67.1 [1.9]	56.2 [2.0]	61.2 [1.5]	66.4 [5.0]	66.1 [6.8]	70.7 [5.5]	67.8 [2.5]
MCD BERT	**91.4 [0.7]**	**90.4 [1.5]**	**90.4 [0.6]**	**90.4 [0.8]**	**71.5 [1.2]**	**67.4 [2.3]**	**59.1 [3.6]**	**62.9 [1.7]**	**68.4 [1.9]**	**68.2 [0.8]**	**69.2 [2.9]**	**68.6 [1.6]**

Table 2. Characteristics of the used datasets: number of the tweets/comments and the embedding architecture used for each of the datasets.

Dataset	**Size**	**MCD LSTM**	**MCD AN**
English	5000	Sentence	Tokenizer
Croatian	8422	Fasttext	Tokenizer
Slovenian	4364	Fasttext	Tokenizer

4. Results

Results consists of three parts: calibration results, prediction performance, and visualization of uncertainty.

4.1. Prediction Performance

The results that compare 4 different models are presented in the Table 1. It can be observed that MCD BERT provide the best results for the all of the 3 data sets. As the MCD BERT is slightly better that BERT we can conclude for the tweets for which BERT is on borderline, multiple predictions can influence decision in the right direction.

With intention to statistically test if MCD BERT could indicate problematic predictions we investigate 1000 test tweets splitting them on the *confused* and *certain*. As the BERT generally provide very extreme predictions the criteria was: the test tweet is confusing if the variance computed from 1000 predictions is greater then 0.1 otherwise it was classified as certain. In the Table 3 two by two contingency results are presented for each of the three languages data sets. The Chi-square test for the English MCD BERT results was found to be very significant with p-value= 2.2e-16. The Chi-square test for BERT model results was found to be less significant with p-value = 1.384e-11. For CRO BERT the Chi-square test was not significant with p-value= 1 so it is clear that we can not classify tweets based just on the probability. On the other hand, for the CRO MCD BERT the criteria based on the variance > 0.1 provide better spit so the Chi-square test become significant with p-value= 8.348e-16. The p-values for the SLO BERT and SLO MCD BERT are 0.0037 and 0.0002 respectively. Also, based on the ratios between mistake and no mistake it can be observed that number of true mistakes in the confused group is high for MCD BERT.

Table 3. Two-by-two contingency table for Certain/Confused vs Mistake yes/no.

Language	**Mistake**	**BERT**		**MCD BERT**	
		Certain	**Confused**	**Certain**	**Confused**
EN	**No**	880	31	891	24
	Yes	71	18	62	23
Ratio		0.08	0.58	0.06	0.95
CRO	**No**	1176	35	1053	152
	Yes	461	14	336	139
Ratio		0.39	0.4	0.31	0.91
SLO	**No**	576	28	537	55
	Yes	241	27	229	51
Ratio		0.42	0.96	0.42	0.92

From those results it can be concluded that the MCD BERT provides better understanding of the how much we can trust our predictions compared to the simple BERT.

5. Conclusions

In practical setting, automatic detection of hate speech not only requires high precision but also prediction uncertainty estimates. In times when social networks suffer from high amount of offensive messages, wrong classifications can damage the minorities, lower the level of democratic debate but also damage the freedom of speech. In technological terms, natural language approaches are witnessing a switch from recurrent neural networks with pretrained word embeddings to large pretrained transformer models, BERT being the best example of this. We introduce the Monte Carlo dropout into attention layers of transformer neural networks as a tool for prediction uncertainty estimation. We demonstrate the methodology on the hate speech detection task.

The results of our empirical evaluation show that MCD can improve BERT results regarding both the prediction performance and uncertainty estimation. For all three languages hate speech datasets, the MCD enhanced BERT and mBERT preformed best. Further, we show that MCD BERT reliability scores provide information on the trusted and dubious predictions. This information can significantly reduce the amount of work in reannotation of questionable cases.

References

Bleich, E. The rise of hate speech and hate crime laws in liberal democracies. *Journal of Ethnic and Migration Studies*, 37(6):917–934, 2011.

Cer, D., Yang, Y., Kong, S.-y., Hua, N., Limtiaco, N., John, R. S., Constant, N., Guajardo-Cespedes, M., Yuan, S., Tar, C., et al. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*, 2018.

Chang, W.-C., Yu, H.-F., Zhong, K., Yang, Y., and Dhillon, I. X-bert: extreme multi-label text classification with bert. *arXiv preprint arXiv:1905.02331*, 2019.

Davidson, T., Warmsley, D., Macy, M., and Weber, I. Automated hate speech detection and the problem of offensive language. In *Eleventh international aaai conference on web and social media*, 2017.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Gal, Y. Uncertainty in deep learning. *University of Cambridge*, 1:3, 2016.

Gal, Y. and Ghahramani, Z. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, pp. 1050–1059, 2016.

Gururangan, S., Dang, T., Card, D., and Smith, N. A. Variational pretraining for semi-supervised text classification. *arXiv preprint arXiv:1906.02242*, 2019.

Izmailov, P., Maddox, W. J., Kirichenko, P., Garipov, T., Vetrov, D., and Wilson, A. G. Subspace inference for bayesian deep learning. *arXiv preprint arXiv:1907.07504*, 2019.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Langley, P. Crafting papers on machine learning. In Langley, P. (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

Ljubešić, N., Fišer, D., and Erjavec, T. The frenk datasets of socially unacceptable discourse in slovene and english. In *International Conference on Text, Speech, and Dialogue*, pp. 103–114. Springer, 2019.

McInnes, L., Healy, J., and Melville, J. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

Miok, K. Estimation of prediction intervals in neural network-based regression models. In *20th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, pp. 463–468, 09 2018.

Miok, K., Nguyen-Doan, D., Robnik-Šikonja, M., and Zaharie, D. Multiple imputation for biomedicaldata using monte carlo dropout autoencoders. In *7th IEEE International Conference on E-Health and Bioengeneering (EHB)*. IEEE, 2019a.

Miok, K., Nguyen-Doan, D., Škrlj, B., Zaharie, D., and Robnik-Šikonja, M. Prediction uncertainty estimation for hate speech classification. In *International Conference on Statistical Language and Speech Processing*, pp. 286–298. Springer, 2019b.

Miok, K., Nguyen-Doan, D., Zaharie, D., and Robnik-Šikonja, M. Generating data using monte carlo dropout. In *2019 IEEE 15th International Conference on Intelligent Computer Communication and Processing (ICCP)*, pp. 509–515. IEEE, 2019c.

Myshkov, P. and Julier, S. Posterior distribution analysis for bayesian inference in neural networks. *Advances in Neural Information Processing Systems (NIPS)*, 2016.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Xu, Y., Qiu, X., Zhou, L., and Huang, X. Improving bert fine-tuning via self-ensemble and self-distillation. *arXiv preprint arXiv:2002.10345*, 2020.

A. Calibration of BAN and BERT

Figure 2 shows how calibration of prediction scores change during training of AN. The red line represents the performance of the fully trained network. It is apparent that additional calibration is necessary – the dotted line represents perfect calibration. Surprisingly, initial training iterations show better calibrated scores. This can be due to the definition of ECE measure: in case that both accuracy and predicted scores are low, this would lead to low ECE value.

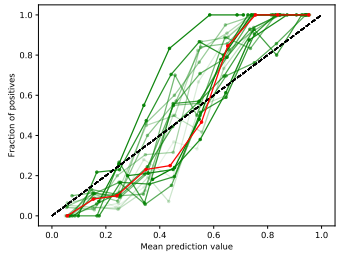


Figure 2. Calibration plot for MCD AN after each epoch (green) based on the validation set of the best performing architecture. The more transparent the calibrations the earlier the training stage (fewer epochs). The final calibration is in red.

In the Tables 4, 5, and 6 calibration results for different calibration approaches of MCD AN are presented: no calibration, isotonic regression, and Platt's method, combined with the adaptive threshold or not. It can be observed that for all three languages both calibration methods improve the ECE score, and Platt's method seems to produce the best calibration scores. Adaptive threshold slightly improves the ECE score for the uncalibrated (raw) results. This is especially true for the Slovenian comments where the ECE score was reduced from the 0.794 to the 0.621. Nevertheless, we can conclude that calibration using adaptive threshold heuristics is beneficial but cannot be compared with the improvements brought by proper calibration techniques.

In order to compare the calibration results for different BAN and BERT architectures, we plotted their ECE scores in Figure 3. It can be observed that calibration methods substantially improve the MCD AN calibration; however, the BERT model is even better calibrated.

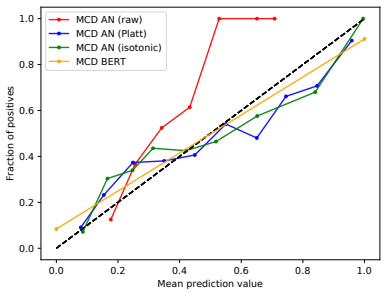


Figure 3. Calibration plots based on English test set performance for BERT and MCD AN architecture using different calibration algorithms.

Table 4. Calibration scores of MCD AN with different calibration approaches on English tweets. The results are presented based on whether they were calibrated and whether the adaptive threshold (AT) was applied

Calibration	AT	Accuracy	F1	ECE
Raw	False	0.83 (0.02)	0.82 (0.03)	0.547
Raw	True	0.83 (0.01)	**0.83** (0.041)	0.54
Isotonic	False	**0.84** (0.01)	0.82 (0.01)	0.230
Isotonic	True	0.83 (0.01)	0.82 (0.02)	0.234
Platt's	False	0.84 (0.02)	0.82 (0.02)	**0.225**
Platt's	True	0.83 (0.01)	0.82 (0.01)	0.232

Table 5. Calibration scores of MCD AN with different calibration approaches on Croatian user news comments. The results are presented based on whether they were calibrated and whether the adaptive threshold (AT) was applied

Calibration	AT	Accuracy	F1	ECE
Raw	False	0.61 (0.02)	0.47 (0.03)	0.681
Raw	True	0.62 (0.02)	**0.50** (0.04)	0.663
Isotonic	False	0.60 (0.01)	0.49 (0.04)	0.206
Isotonic	True	0.61 (0.01)	0.50 (0.03)	0.206
Platt's	False	0.61 (0.02)	0.48 (0.02)	0.198
Platt's	True	**0.62** (0.02)	0.49 (0.02)	**0.197**

Table 6. Calibration scores of MCD AN with different calibration approaches on Slovenian Facebook comments. The results are presented based on whether they were calibrated and whether the adaptive threshold (AT) was applied

Calibration	AT	Accuracy	F1	ECE
Raw	False	0.59 (0.01)	0.33 (0.05)	0.794
Raw	True	0.59 (0.02)	0.48 (0.05)	0.621
Isotonic	False	0.58 (0.02)	0.49 (0.03)	0.212
Isotonic	True	0.58 (0.02)	**0.49** (0.03)	0.213
Platt's	False	0.58 (0.03)	0.48 (0.02)	0.206
Platt's	True	**0.59** (0.02)	0.47 (0.04)	**0.204**

B. Additional reliability graphs

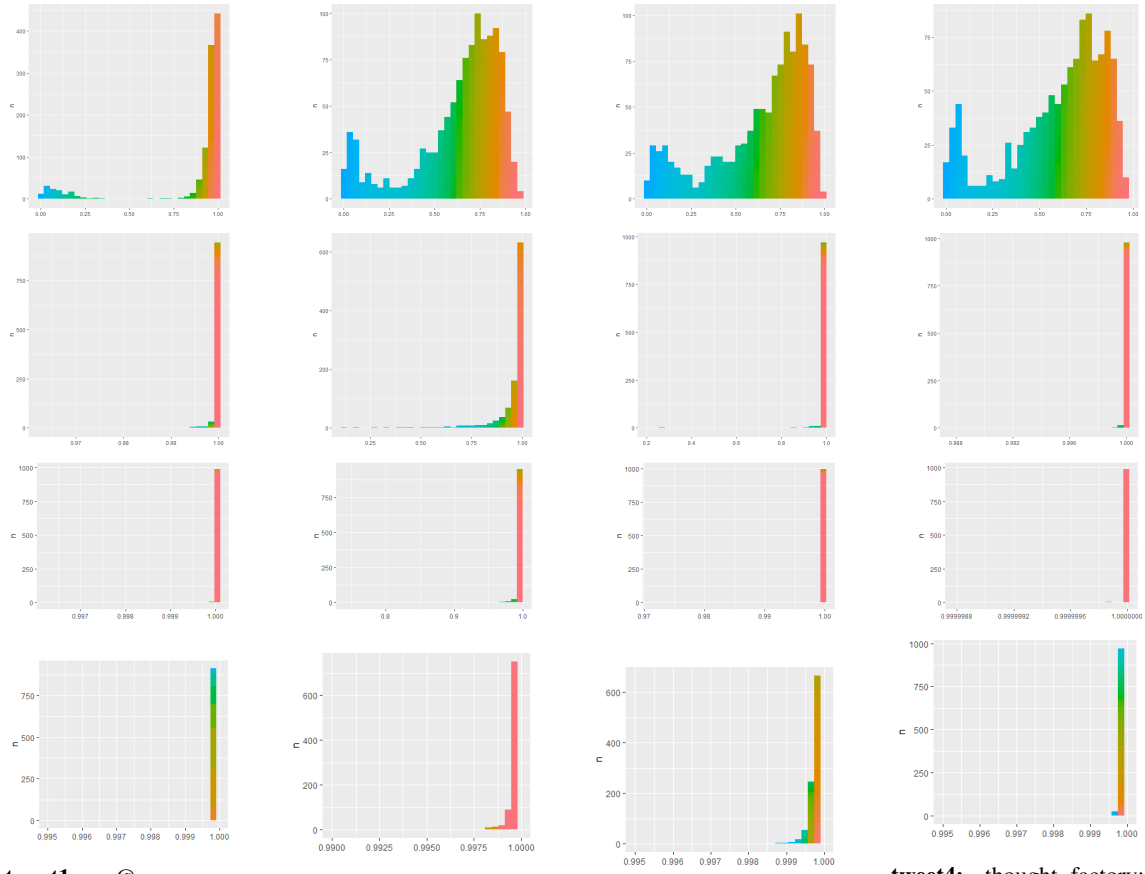
B.1. Visualization of Uncertainty

Obtaining multiple predictions for a specific test tweet can improve understanding of the final prediction. The mean of the distribution is used to estimate the probability and the variance informs us about the spread and certainty of the prediction. We can inspect the actual distribution of prediction scores with histogram plots as demonstrated on Figures 4 and 6 for a few correctly classified instances and on Figures 5 and 7 for a few misclassified instances.

Histograms presented in the Figures 4 and 5 for English tweets and Figures 6 and 7 for Croatian comments visually display the prediction certainty of the specific tweet or comment. It can be observed that results for BERT are going to the extremes, especially for the predictions when model seems to be sure. Also, it can be observed that AN architecture with 10% of dropout provide the similar spread of values as the BERT. On the other hand, introducting 30 % of dropout in AN for examples where attention network model is not sure will influence the spread and make the prediction more uncertain.

Apart from visualization text tweet separately the multiple prediction provide opportunity for understanding the contextual dependencies with the other test tweets. Following (Miok et al., 2019b) we visualize the embeddings of the output. The key idea of the visualization can be summarized as follows. First, 1000 samples are obtained for each prediction. The space of such distributions across individual test-set texts is next *embedded* into two dimensions by using Uniform Manifold Projections method (McInnes et al., 2018). this way, a two dimensional space corresponding to the initial 1000 dimensional space of predictions is obtained. Next, Gaussian kernel estimation is used to identify equivalent regions, which are connected with closed curves. Finally, the shapes and sizes of individual predictions are adapted based on the classification error and certainty of a given prediction. The goal of using such visualization is to discover potentially larger structures within the space of emitted probabilities, potentially offering insights into the given probabilistic neural network's drawbacks and limitations. The results of such visualization are shown in Figures 8 and 9. In Figure 8 the plot displays the position of the certain and uncertain test tweets in the latent space while in the Figure 9 the differences based on the mean probability are displayed.

It can be observed that (Figure 8) after only a few epoch of training, majority of the prediction are in the middle layer (yellow) that corresponds to the predictions that are uncertain (high spread of the whole predictive distribution). On the contrary, in Figure 9, where the same learning setting was considered for 50 epochs, the probability space distinctly separates into two main *components*, indicating that there are predictions for which the neural network is certain (and were indeed correct), however for some predictions, especially related to the instances that are not hate speech, the network is less certain (albeit still correct). The two examples demonstrate how the space of probabilities *separates* into distinct components once the neural network is trained. The visualizations also indicate that some of the instances are more problematic than others, potentially facilitating the debugging process for a developer (and inspection of convergence).



tweet1: @user you might be a libtard if... #libtard #sjw #liberal #politics

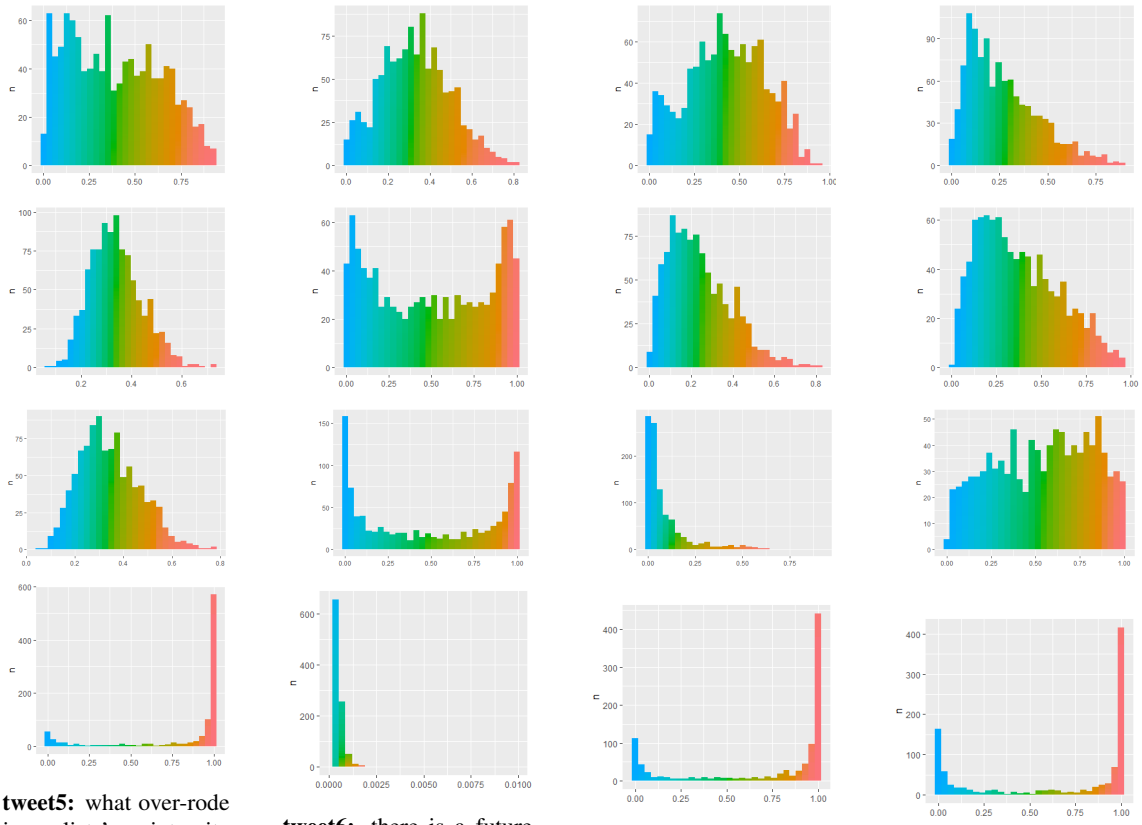
tweet2: #paid #kkk to #fabricate #stories to push the producers #narrative #cancel

tweet3: carl paladino, trump ally, wishes obama dead of mad cow disease

tweet4: thought factory: left-right polarisation! #trump #uselections2016 #leadership #politics #brexit #blm

Figure 4. English data set - Comparing the multiple prediction distributions for MCD LSTM (first row), MCD AN with 30% dropout (second row), MCD AN with 10% dropout (third row) and MCD BERT (fourth row) for 4 test tweets where hate speech was **correctly** predicted. Note that the x axis showing predicted probability distributions are different for each tweet. Results of BERT are concentrated in much narrower interval compared to MCD LSTM and MCD AN.

Bayesian BERT for Trustful Hate Speech Detection



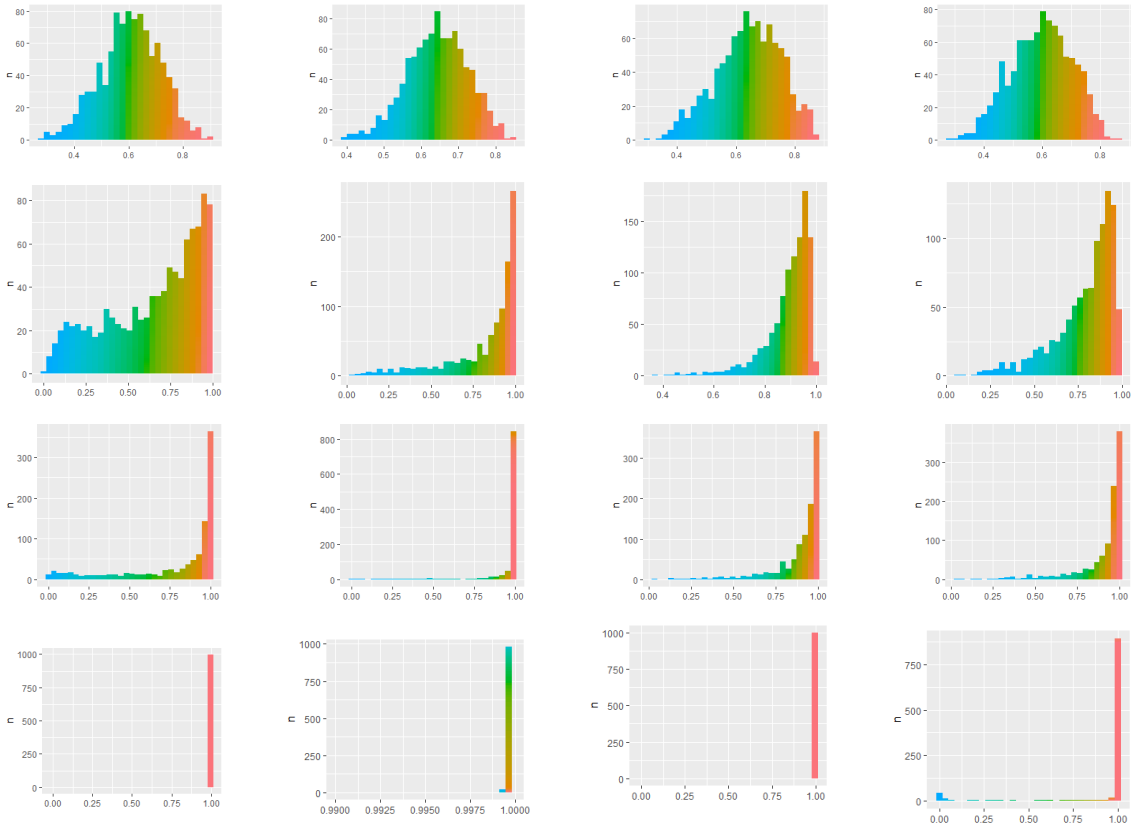
tweet5: what over-rode journalists' integrity was greed and ambition, along with a total absence of courage (true label = hate speech)

tweet6: there is a future nina and right now you are pretty much throwing yours away. (true label = non hate speech)

tweet7: @user @user someone really respects eu guidelines!!! (true label = non hate speech)

tweet8: the scars left by us waime camps mt @user (true label = hate speech)

Figure 5. English data set - Comparing the multiple prediction distributions for MCD LSTM (first row), MCD AN with 30% dropout (second row), MCD AN with 10% dropout (third row) and MCD BERT (fourth row) for 4 test tweets where hate speech was not clearly predicted. Note that the x axis showing predicted probability distributions are different for each tweet. Results of BERT are concentrated in much narrower interval compared to MCD LSTM and MCD AN.



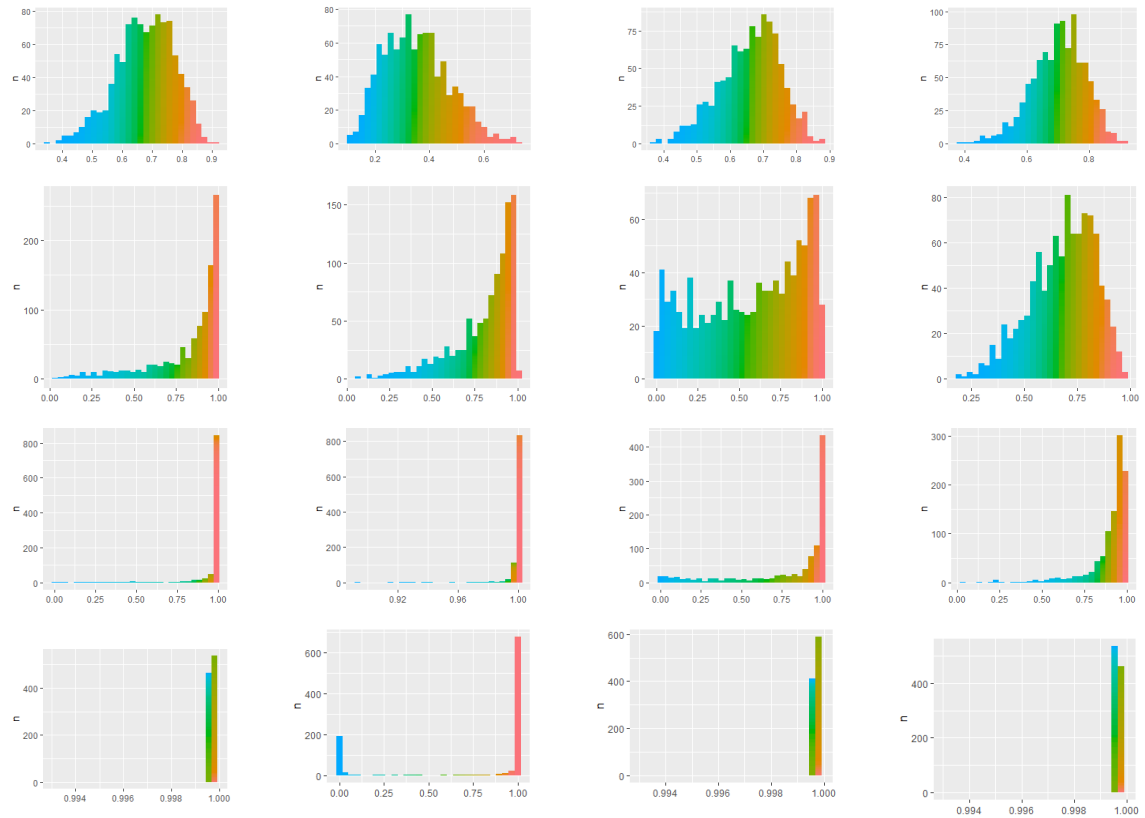
81 - kamo sreće da jeste...al mi u HR nismo imali pojam ”rodjo” dok se vi niste naselili ovdje

17 - Drazi Mihailovicu je podignut spomenik i u Usa.Kada vas Pavelic dobije spomenik,bilo gde u svetu,onda se javite.Istina boli vise od batina,zar ne?

2 - prvo treba najuriti pola sabora na celu sa cedom pupijo . poz... od HOSa ZDS

24 - I mi srbi volimo hrvatice, jedva cekam vikend i njihov dolazak u Bg.

Figure 6. Croatian data set - Comparing the multiple prediction distributions for MCD LSTM (first row), MCD AN with 10% dropout (second row), MCD AN with 30% dropout (third row) and MCD BERT (fourth row) for 4 test tweets where hate speech was not clearly predicted. Note that the x axis showing predicted probability distributions are different for each tweet. Results of BERT are concentrated in much narrower interval compared to MCD LSTM and MCD AN.



18 - Eh vi Hrvati vas problem je sto vi sve nas Srbe smatrate cetnicima,ima nas i komunista,a i pacifista,ja sam pacifisticki nastrojen zbog toga kazem Pruzimo Miru sansu,i ne dirajmo dvojezicne table jer su one simbol jedinstva SrboHrvata! (true label = non hate speech)

31 - mali sloba na reveru nosi zastavu srbije a naš milanče je bio neodlučan– (true label = non hate speech)

56 - Rubinet@ / Pa bas i da ste slobodni i niste.Jednog gospodara Beograd zamenili ste sa novim Brisel,kao kroz citavu vasu povijest.Hej jos se i dice time da imaju gospodara ! (true label = hate speech)

19 - Posledica suživota sa nama?ahhhhahhh ćuti bre konjušaru (true label = hate speech)

Figure 7. Croatian data set - Comparing the multiple prediction distributions for MCD LSTM (first row), MCD AN with 30% dropout (second row), MCD AN with 10% dropout (third row) and MCD BERT (fourth row) for 4 test tweets where hate speech was not clearly predicted. Note that the x axis showing predicted probability distributions are different for each tweet. Results of BERT are concentrated in much narrower interval compared to MCD LSTM and MCD AN.

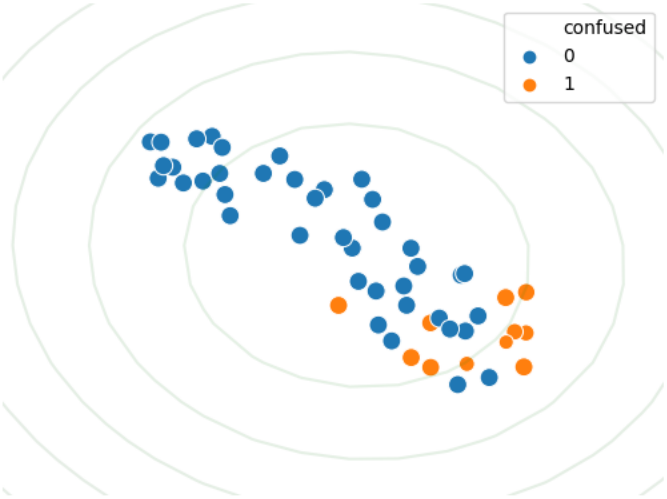


Figure 8. Visualization of the 100 test tweets in two dimensions. Tweets that are found to be certain are colored in blue (0) while tweets that are confused in orange (1). It can be observed that uncertain tweets get clustered.

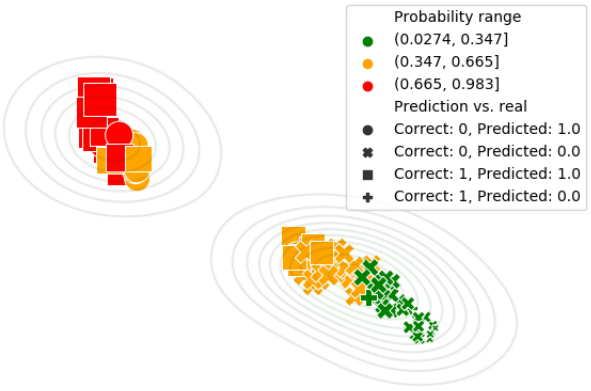


Figure 9. Visualization of the outcome probability space for 100 tweets from the test set. The test tweets are colored in the green, yellow and red depending to which interval belongs the mean probability of the 1000 predictions. It can be observed that the predictions with very high confidence form an isolated part of the probability space.

Appendix E: Propositionalization and Embeddings: Two Sides of the Same Coin

Machine Learning (2020) 109:1465–1507
https://doi.org/10.1007/s10994-020-05890-8

Propositionalization and embeddings: two sides of the same coin

Nada Lavrač[1,2] · Blaž Škrlj[3] · Marko Robnik-Šikonja[4]

Received: 15 February 2019 / Revised: 29 May 2020 / Accepted: 8 June 2020 / Published online: 28 June 2020

Abstract
Data preprocessing is an important component of machine learning pipelines, which requires ample time and resources. An integral part of preprocessing is data transformation into the format required by a given learning algorithm. This paper outlines some of the modern data processing techniques used in relational learning that enable data fusion from different input data types and formats into a single table data representation, focusing on the propositionalization and embedding data transformation approaches. While both approaches aim at transforming data into tabular data format, they use different terminology and task definitions, are perceived to address different goals, and are used in different contexts. This paper contributes a unifying framework that allows for improved understanding of these two data transformation techniques by presenting their unified definitions, and by explaining the similarities and differences between the two approaches as variants of a unified complex data transformation task. In addition to the unifying framework, the novelty of this paper is a unifying methodology combining propositionalization and embeddings, which benefits from the advantages of both in solving complex data transformation and learning tasks. We present two efficient implementations of the unifying methodology: an instance-based PropDRM approach, and a feature-based PropStar approach to data transformation and learning, together with their empirical evaluation on several relational problems. The results show that the new algorithms can outperform existing relational learners and can solve much larger problems.

Keywords Inductive logic programming · Relational learning · Propositionalization · Embeddings · Knowledge graphs

Editors: Dimitar Kazakov and Filip Železny.

✉ Blaž Škrlj
blaz.skrlj@ijs.si

Extended author information available on the last page of the article

1 Introduction

Data preprocessing for machine learning is a great challenge for a data scientist faced with large quantities of data in different forms and sizes. Most of the modern data processing techniques enable data fusion from different data types and formats into a single table data representation, which is expected by standard machine learning techniques including rule learning, decision tree learning, support vector machines (SVMs), deep neural networks (DNNs), etc. The key element of the success of modern data transformation methods is that similarities of original instances and their relations are encoded as distances in the target vector space.

Two of the most prominent data transformation approaches outlined in this paper are propositionalization and embeddings. While propositionalization (Kramer et al. 2001; Železný and Lavrač 2006) is a well known data transformation technique used in relational learning (RL) and inductive logic programming (ILP) (Muggleton 1992; Lavrač and Džeroski 1994; De Raedt 2008), embeddings (Mikolov et al. 2013; Wu et al. 2018) have only recently been recognized by RL and ILP researchers as a powerful technique for preprocessing relational and complex structured data. In the relational learning context of this paper, both approaches take as input a relational data set (e.g., a given relational database) and transform it into a single data table format, which is then used as an input to a propositional learning algorithm of choice.

The first aim of this paper is to present a unifying survey of propositionalization and embedding data transformation approaches. While both approaches aim at transforming data into a tabular data format, the approaches use different terminology and task definitions, claim to have different goals, and are used in very different contexts. This paper contributes an improved understanding of these data transformation techniques by presenting a unified terminology and definitions, by explaining the similarities and differences of the two definitions as variants of a unified complex data transformation task, by exploring the apparent differences between the two approaches, and by outlining some of their advantages and disadvantages.

In addition to the unifying survey, the main novelty of this paper is a unifying methodology that combines propositionalization and embeddings, which benefits from the advantages of both in solving complex data transformation and learning tasks. The unifying methodology resulted in two new pipelines, PropDRM and PropStar, which implement an instance-based and a feature-based approach to data transformation and learning, respectively. Both approaches are computationally efficient and can successfully solve much larger tasks than the existing relational learning approaches. We made their code publicly available.

The paper starts by motivating the need for transforming heterogeneous relational data into a tabular format in Sect. 2. Section 3 introduces the data transformation approaches in the context of information representation levels proposed by Gärdenfors (2000). Section 4 presents the related work, focusing on selected propositionalization and embeddings methods relevant to the relational learning context of this paper. Section 5 presents a unifying framework for propositionalization and embeddings, allowing for the analysis of characteristic properties of these data transformation approaches. Section 6 proposes a unifying methodology that combines propositionalization and embeddings, which benefits from the advantages of both, and presents two implementations of the proposed unifying framework: an instance-based embedding approach PropDRM based on the existing Deep Relational Machines (DRM) (Srinivasan et al. 2019; Lodhi 2013), followed by a novel feature-based

embedding approach PropStar proposed in this paper, using the StarSpace entity embedding approach (Wu et al. 2018). Experimental evaluation of the proposed implementations is presented in Sect. 7. The paper concludes by a summary and some ideas for future work in Sect. 8.

2 Motivation

Machine learning is the key enabler for computer systems to progressively improve their performance when helping humans to solve difficult problem solving tasks. Nevertheless, current machine learning approaches only come half-way in helping humans, as humans still have to formulate the problem and prepare the data in the form that is best suited to the powerful machine learning algorithms.

Most of the best performing machine learning algorithms, like Support Vector Machines (SVMs) or deep neural networks, assume numeric data and outperform symbolic approaches in terms of predictive performance, efficiency, and scalability. The dominance of numeric algorithms started in 1980s with the advent of backpropagation and neural networks (Rumelhart et al. 1986), continued in late 1990s and early 2000s with SVMs (Cortes and Vapnik 1995), and finally reached the current peak with deep neural networks (Goodfellow et al. 2016). Deep neural networks are currently considered the most powerful learners for solving many of previously unsolvable learning problems in computer vision (face recognition rivals humans' performance), game playing (a program has beaten a human champion in the game of Go), and natural language processing (successful automatic speech recognition and machine translation).

While the most powerful machine learning approaches are numeric, humans perceive and describe real-world problems mostly in symbolic terms, using various data representation format, such as graphs, relations, texts or electronic health records, all involving discrete representations. However, if we are to harness the power of successful numeric deep learning approaches for discrete learning problems, discrete data should be transformed into a form suitable for numeric learning algorithms. The viewpoint of addressing real-world problems as numeric has a rationale even for discrete domains, as many symbolic learners perform generalizations based on object similarity. For example, in graphs, nodes can represent similar entities or have connections with similar other nodes; in text, words can appear with similar contexts or play the same role in sentences; in medicine, patients may have similar symptoms or similar disease histories. Such similarities are used by numerous machine learning algorithms to generalize and learn, including classical bottom-up learning approaches such as hierarchical clustering, as well as symbolic learners adapted to top-down induction of clustering trees (Blockeel et al. 1998). If we want to exploit the power of modern machine learning algorithms, like SVMs and deep neural networks, to process the inherently discrete data, one has to transform discrete data into (numeric) vectors in such a way that similarities between objects are preserved and encoded as distances in the transformed (numeric) space.

Contemporary preprocessing approaches that prepare numeric vector data for machine learning algorithms are called *embeddings*. Nevertheless, as demonstrated in this paper, symbolic data transformations, as ancestors of the contemporary embedding approaches, remain relevant: the role of *propositionalization*, a symbolic approach to relational data transformation into feature vectors, is not only to enable contemporary machine

learning algorithms to induce better predictive models, but to allow descriptive data mining approaches to discover interesting human-comprehensible patterns in symbolic data.

As this paper demonstrates, albeit propositionalization and embeddings represent different types of data transformations, these approaches actually represent the *two sides of the same coin*. The main unifying element they have in common is that they transform the data into a vector format and encode the relations between objects in the original space as distances in the new vector space.

3 Data transformations and information representation levels

As this section will show, we consider data transformations as a particular subprocess of data preprocessing. Data preprocessing aims to handle missing attribute values, control out-of-range values and impossible attribute-value combinations, or handle noisy or unreliable data, to name just some of the types of data irregularities addressed in processing real-life data. Data preprocessing may include data cleaning, instance selection, normalization, feature engineering (feature extraction and/or feature construction), data transformation, feature selection, etc. The result of data preprocessing is the final training set, which is used as input to a machine learning algorithm.

Data preprocessing can be manual, automated, or semi-automated. We focus on automated transformations of data, present in heterogeneous types and formats, into a uniform tabular data representation. We refer to this specific automated data preprocessing task as *data transformation*, and define it as follows.

Definition 1 (*Data transformation*) *Data transformation* is a step in the data preprocessing task that automatically transforms the input data and the background knowledge into a uniform tabular representation, where each row represents a data instance, and each column represents one of the dimensions in a multi-dimensional feature space.

In the above definition, we decided to distinguish between *data* and *background knowledge*. This is an intentional decision, although it could be argued that in some settings, we could refer to both as data. Let us provide an operational distinction between data and background knowledge. *Data* is considered by the learner as the target data from which the learner should learn a model (e.g., a classifier in the case of class labeled data) or a set of descriptive patterns (e.g., a set of association rules in the case of unlabeled data). *Background knowledge* is any additional knowledge used by the learner in model or pattern construction from the target data. Simplest forms of background knowledge define hierarchies of features (attribute values), such as color *green* being more general than *light green* or *dark green*. More complex background knowledge refers to any other declarative prior domain knowledge, such as knowledge encoded in relational databases, knowledge graphs or domain specific taxonomies and ontologies, such as the Gene Ontology, in its 2020-05-02 release including 44,508 GO terms, 7,765,270 annotations, 1,464,358 gene products and 4,593 species.

This data transformation setting is applicable in various data science scenarios involving relational data mining, inductive logic programming, text mining, graph and network mining as well as tasks that require fusion of data of a variety of data types and formats and their transformation into a joint data representation formalism.

3.1 Information representation levels

As currently the most powerful machine learning (ML) algorithms take as input numeric representations, users of ML algorithms tend to transform other forms of human knowledge into the numeric representation space. Interestingly, even if this is countering a standard RL and ILP viewpoint, this is true also for symbolic representations, which are currently used to store most of the human knowledge.

The distinction between the symbolic and numeric representation space mentioned above can be further clarified in terms of the *levels of cognitive representations*, introduced by Gärdenfors (2000), i.e. the neural, spatial and symbolic representation levels. In his theory, Gärdenfors assumes that when modeling cognitive systems in terms of information processing, all three levels are connected: starting from the sensory inputs at the lowest neural representation level, resulting in spatial representations at the middle conceptual spaces level, up to symbolic representations at the level of language.

Neural This representation level corresponds to the sub-conceptual connectionist level. At this level, information is represented by activation patterns in densely connected networks of primitive units. This enables concepts to be learned from the observed data by modifying the connection weights between the units.

Spatial This representation level is modeled in terms of Gärdenfors' conceptual spaces. At this level, information is represented by points or regions in a conceptual space built upon some dimensions that represent geometrical, topological or ordinal properties of the observed objects. In spatial representations, the similarity between concepts is represented in terms of the distances between the points or regions in a multidimensional space, where concepts are learned by modeling the similarity between the observed objects.

Symbolic At this representation level, information is represented by the language of symbols (words), where the meaning is internal to the representation itself (i.e. symbols have meaning only in terms of other symbols, while their semantics is grounded in the spatial level), and concepts are learned by symbolic generalization rules.

From the perspective of this paper, the above levels of cognitive representations introduced by Gärdenfors (2000) provide a theoretical ground to separate the learning approaches as well as the data transformation approaches into three categories based on the levels of their output representation space: neural, spacial and symbolic. However, given the scope of this paper, we do not consider *neural transformations*, and focus only on two data transformation types:

- *symbolic transformations*, in this paper referred to as *propositionalization*, denoting data transformations into a symbolic representation space, and
- *numeric transformations*, in this paper referred to as *embeddings*, denoting data transformations into a spatial representation space.

These two data transformation approaches are briefly introduced below, and further described in the related work (Sect. 4).

3.2 Transformations into symbolic representation space

The past decades of machine learning were characterized by symbolic learning, where the result of a machine learning or data mining algorithm was a predictive model of a set of patterns described in a symbolic representation language, resulting in symbolic human-understandable patterns and models. Symbolic machine learning approaches include rule learning (Michalski et al. 1986; Clark and Niblett 1989), decision tree learning (Quinlan 1986) and learning logical representations by relational learning and inductive logic programming (ILP) algorithms (Muggleton 1992; Lavrač and Džeroski 1994; De Raedt 2008).

To be able to apply a symbolic learner, the data is typically transformed into a single tabular data format, where each row represents a single data instance, and each column represents an attribute or a feature. Such transformation into symbolic vector space (i.e. a symbolic data table format) is well known in the ILP and relational learning community, where it is referred to as *propositionalization*. Propositionalization approaches are presented in Sect. 4.2.

3.3 Transformations into numeric representation space

In the last 20 years we have been witnessing increasing dominance of statistical machine learning and pattern-recognition methods, including neural network learning (Rumelhart and McClelland 1986), Support Vector Machines (SVMs) (Vapnik 1995; Schölkopf and Smola 2001), random forests (Breiman 2001), and boosting (Freund and Schapire 1997). These statistical approaches are quite different from the symbolic approaches mentioned in Sect. 3.2, however there are many approaches that cross these boundaries, including e.g., the CART decision tree learning algorithm (Breiman et al. 1984), the Bump hunting rule learning algorithm (Friedman and Fisher 1999), which are firmly based in statistics. Moreover, ensemble techniques such as boosting (Freund and Schapire 1997), bagging (Breiman 1996) or random forests (Breiman 2001) also combine the predictions of multiple logical models on a sound statistical basis (Schapire et al. 1998; Mease and Wyner 2008; Bennett et al. 2008). All these are also considered to belong to the family of statistical learning approaches.

To be able to apply a statistical learner, the data is typically transformed into a single tabular data format, where each row represents a single data instance, and each column is a numeric attribute or a numeric feature, with some predefined range of numeric values. Such transformation into numeric vector space (i.e. a numeric data table format) is well known in the deep learning community, where it is referred to as *embedding*. Approaches to embedding relational structures are presented in Sect. 4.3.

4 Related work

In this section we first outline various transformation methods in Sect. 4.1, followed by a more detailed description of the data transformation methods relevant for the context of relational learning, i.e. propositionalization and embeddings, in Sects. 4.2 and 4.3, respectively.

4.1 Outline of data transformation methods

While there are many algorithms for transforming data into a spatial representation, it is interesting that recent approaches rely on deep neural networks, thereby harnessing the neural representation level as the means to transform symbolic representations into the spatial representation. Below we list the main types of approaches that perform transformations between representations.

Community detection and graph traversal methods. Many complex data sets can be represented as graphs, where nodes represent data instances and edges represent their relations. Graphs can be homogeneous (consisting of a single type of nodes and relations) or heterogeneous (consisting of different types of nodes and relations). To encode a graph in a tabular form by preserving the information about the relations, various graph encoding techniques were developed, such as propositionalization via random walk graph traversal, representing nodes via their neighborhoods and communities (Plantié and Crampes 2013). These approaches are frequently used for data fusion in mining heterogeneous information networks. Neural network approaches (presented below) are also very competitive as means for encoding graphs.
Matrix factorization methods. When data is not explicitly presented in the form of relations but the relations between objects are implicit, given by a similarity matrix, the objects can be encoded in a numeric form using matrix factorization. As an example take Latent Semantic Analysis used in text mining, which factorizes a word similarity matrix to represent words in a vector form. Another example is factorization of graph adjacency matrices. These types of embeddings were largely superseded by deep neural networks which, instead of observing similarity between different objects, construct a prediction task and forecast similarity. For example, for text, given a word, the word-2vec embedding method (Mikolov et al. 2013) predicts words in its neighborhood.
Propositionalization methods are used to get tabular data from multirelational databases as well as from a mixture of tabular data and background knowledge in the form of logic programs or networked data, including ontologies. These transformations were mostly developed within the Inductive Logic Programming and Relational Learning community, and are still actively researched and used. Propositionalisation methods do not perform dimensionality reduction and are most often used with data mining and symbolic machine learning algorithms. We discuss these methods in Sect. 4.2.
Neural networks based methods. In neural networks the information is represented by activation patterns in interconnected networks of primitive units. This enables that concepts are gradually learned from the observed data by modifying the connection weights between the hierarchically organized units. These weights can be extracted from neural networks and used as a spatial representation that transforms relations between entities into distances. Recently, this approach became a prevalent way to build representation for many different types of entities, e.g., texts, graphs, electronic health records, images, relations, recommendations, etc. In Sect. 4.3 we describe the data types and approaches, which are capable of embedding relational structures and are therefore most relevant for the context of this paper. These include knowledge graph embeddings (presented in Sect. 4.3.1), entity embeddings capable of forming (both supervised and unsupervised) representations based on the similarity of entities (presented in Sect. 4.3.2), and Deep Relational Machines methodology that links symbolic representations to deep neural networks (presented in Sect. 4.3.3).

Other embedding methods. Other forms of embeddings were developed by different communities that observed the need to better represent the (symbolic) data. For example, Latent Dirichlet Allocation (LDA) (Blei et al. 2003) used in text analysis learns distributions of words for different topics. These distributions can be used as an effective embedding for words, topics, and documents. Feature extraction methods form a rich representation of instances by projecting them into a high dimensional space (Lewis 1992). Another example of (implicit) transformation into high dimensional space is the kernel convolutional approach proposed by Haussler (1999), which introduces the idea that kernels can be used for discrete structures by iteratively applying convolution and kernels to smaller parts of the data structure. Convolutional kernels exist for sets, graphs, trees, strings, logical interpretations, and relations (Cumby and Roth 2003). This allows methods such as SVM or Gaussian Processes to work with relational data. Most of these embeddings are recently superseded or merged with neural networks.

All the above approaches perform data transformations from different data formats to a single table representation. However, their underlying principles are different: while factorization and neural embeddings perform dimensionality reduction, resulting in lower-dimensional feature vector representations capturing the semantics of the data, propositionalization results in a vector representation using relational features with a higher generalization potential than the features used in the original data representation. Note that there exist also other approaches to data transformation and fusion, including HINMINE (Kralj et al. 2018), metapath2vec (Zhu et al. 2018) and OhmNet (Žitnik and Leskovec 2017), which are out of the main scope of this paper.

4.2 Propositionalization

In propositionalization, relational feature construction is the most common approach to data transformation. LINUS (Lavrač et al. 1991) was one of the pioneering propositionalization approaches using automated relational feature construction. LINUS was restricted to generation of features that do not allow recursion and existential local variables, which means that the target relation cannot be many-to-many and self-referencing. The second limitation was more serious: the queries could not contain joins (conjunctions of literals). The LINUS descendant SINUS (Lavrač and Flach 2001) incorporates more advanced feature construction techniques inspired by 1BC (Flach and Lachiche 1999). The LINUS approach had many followers, including relational subgroup discovery system RSD (Železný and Lavrač 2006), which is outlined also in the list of propositionalization approaches below. Alternatives to relational feature construction include the construction of aggregation queries.

In this section we first clearly define the distinction between attributes and features, followed by an outline of selected propositionalization approaches and of the specific Wordification approach used in the algorithms developed in this work.

4.2.1 Features

To be able to apply a symbolic propositional learner, the data should be represented in a single table data format, where each row represents a single data instance, and each column represents an attribute or a feature. For the sake of clarity, let us distinguish between *attributes* and *features* below.

Attributes that describe the data instances can be either numeric variables (with values like 7 or 1.5) or nominal/discrete variables (with values like *red* or *female*). In contrast to attributes, a *feature* describes the presence or absence of some property of an instance. As a result, features are always Boolean-valued (values *true* or *false*). For example, for attribute *gender* with values *female* and *male*, two separate features can be constructed: f_1: *gender=female* and f_2: *gender=male*, and only one of these features is assumed to be *true* for an individual data instance. Note that features are different even from binary-valued attributes: e.g., for a binary attribute a_i with values *true* and *false*, there are two corresponding features: f_3: $a_i = true$ and f_4: $a_i = false$. Furthermore, features can test a value of a single attribute, like $a_j > 3$, or they can represent complex logical and numerical relations, integrating properties of multiple attributes, like f_5: $a_k < 2 \cdot (a_j - a_i)$.

Previous feature types are referred to as *propositional features*. On the other hand, *relational features* relate the values of different attributes to each other. In the simplest case, for example, they test for the equality or inequality of the values of two attributes of the same type, such as *Length* and *Height*. More complex relational features can use the background relations, e.g., f_6: *adjacent(NodeX, NodeY)*. Even more advanced, relational features can introduce new variables. For example, if relations are used to encode a graph, a relational feature such as f_7: *color(CurrentNode, blue)* $\wedge$ *link(CurrentNode, NewNode)* $\wedge$ *color(NewNode, red)*, can introduce a new variable *NewNode* to subsequently test whether there exists a previously not visited node in the graph that is colored red.

Take a simple toy trains example learning problem illustrated in Appendix A, and two complex relational features describing trains:

f_8: hasCar(T,C) $\wedge$ carLength(C,short) $\wedge$ carRoof(C,peaked)

f_9: hasCar(T,C1) $\wedge$ carLength(C1,short) $\wedge$ hasCar(T,C2) $\wedge$ carRoof(C2,peaked)

Feature f_8 is a single complex relational feature, while f_9 contains two distinct relational features. Formally, a feature is defined as a minimal set of literals such that it introduces at most one local (i.e. existential) variable in the feature set composing the relational feature.

The main point of relational features is that they localize variable sharing: this can be made explicit by naming the features:

f_{10}: hasShortCar(T) $\leftarrow$ hasCar(T,C) $\wedge$ clength(C,short)

f_{11}: hasPeakedroofCar(T) $\leftarrow$ hasCar(T,C) $\wedge$ carRoof(C,peaked)

The propositionalization approach to relational learning captures exactly this idea: generating complex features, such as f_8, f_{10} and f_{11}, which will allow multi-relational data representation of properties of target instances (such as trains *T*) through representations of properties of their components (such as cars *C*). Selected propositionalization approaches, which use complex feature construction in the automated multi-relational data transformation process are outlined below.

4.2.2 Outline of selected propositionalization algorithms

Below we outline a selection of propositionalization approaches, while an interested reader can find extensive overviews of different feature construction approaches in the work of Kramer et al. (2001) and Krogel et al. (2003).

Relaggs (Krogel and Wrobel 2001) stands for *rel*ational *agg*regation. It is a propositionalization approach that takes the input relational database schema as a basis for a declarative bias, using optimization techniques usually used in relational databases (e.g.,

indexes). The approach employs aggregation functions in order to summarize non-target relations with respect to the individuals in the target table.

1BC (Flach and Lachiche 1999) strives to enable the propositional naive Bayes classifier to handle relational data. It does so by a transformation in which a set of first-order conditions is generated and then used as attributes in the naive Bayes classifier. The transformation, however, is done in a dynamic manner, as opposed to standard propositionalization, which is performed as a static step of data preprocessing. This approach is extended by 1BC2 (Lachiche and Flach 2003), which allows distributions over sets, tuples, and multisets, thus enabling the naive Bayes classifier to consider also structured individuals.

Tertius (Flach and Lachiche 2001) is a top-down rule discovery system, incorporating first-order clausal logic. The main idea is that no particular prediction target is specified beforehand, hence Tertius can be seen as an ILP system that learns rules in an unsupervised manner. Its relevance for this survey lies in the fact that Tertius encompasses 1BC, i.e. relational data is handled through 1BC transformation.

RSD (Železný and Lavrač 2006) is a relational subgroup discovery algorithm composed of two main steps: the propositionalization step and the (optional) subgroup discovery step. The output of the propositionalization step can be used also as input to other propositional learners. RSD effectively produces an exhaustive list of first-order features that comply with the user-defined mode constraints, similar to those of Progol (Muggleton 1995) and Aleph (Srinivasan 2007). Furthermore, RSD features satisfy the connectivity requirement, which imposes that no feature can be decomposed into a conjunction of two or more features. Mode declarations define the algorithm's syntactic bias, i.e. the space of possible features.

HiFi (Kuželka and Železný 2008) is a propositionalization approach that constructs first-order features with hierarchical structure. Due to this feature property, the algorithm performs the transformation in polynomial time of the maximum feature length. Furthermore, the resulting features are the shortest in their semantic equivalence class. The algorithm is shown to perform several orders of magnitude faster than RSD for higher feature lengths.

RelF (Kuželka and Železný 2011) is the most relevant of the algorithms in the Tree-Liker software (Kuželka and Železný 2011). It constructs a set of tree-like relational features by combining smaller conjunctive blocks. RelF preserves the monotonicity of feature reducibility and redundancy (instead of the typical monotonicity of frequency), which allows the algorithm to scale far better than other state-of-the-art propositionalization algorithms.

Cardinalization (Ahmed et al. 2015) is specifically designed to enable more than just categorical attributes in propositionalization. Specifically, it can handle a threshold on numeric attribute values and a threshold on the number of objects satisfying the condition on the attribute simultaneously. Cardinalization can be seen as an implicit form of discretization. While in discretization one sets a threshold on a numeric attribute and see how many objects satisfy the threshold later, and the cardinality follows implicitly from the attribute value threshold; on the other hand, in cardinalization, we set a threshold on the cardinality, and let an attribute-value learner decide where the threshold value on the numerical attribute should lie. Hence, Cardinalization allows for context-aware discretization. Quantiles (Ahmed et al. 2015) is a variant of Cardinalization. Instead of choosing an absolute number as cardinality threshold, Quantiles uses a relative number.

CARAF (Charnay et al. 2015) approaches the problem of large relational feature search space by aggregating base features into complex compounds, which makes CARAF

similar to Relaggs. Complex aggregates run the risk of overfitting. While Relaggs tackles this problem by restricting itself to relatively simple aggregates, the distinguishing feature of CARAF is that instead it incorporates more complex aggregates into a random forest, which ameliorates the overfitting effect.

Aleph (Srinivasan 2007) is the most popular ILP algorithm and is actually an ILP toolkit with many modes of functionality: learning of theories, feature construction, incremental learning, etc. Aleph uses mode declarations to define the syntactic bias. Input relations are Prolog clauses, defined either extensionally or intensionally. Aleph's feature construction functionality also means it is a propositionalization approach.

Wordification (Perovšek et al. 2013, 2015) is a propositionalization method inspired by text mining that can be viewed as a transformation of a relational database into a corpus of text documents. The distinguishing property of Wordification is its efficiency when used on large relational data sets and the potential for using text mining approaches on the transformed propositional data. While most of the outlined propositionalization algorithms construct complex relational features including variables in the arguments of relational features, Wordification constructs simple, easily interpretable features that are treated as 'words' in the transformed Bag-Of-Words representation. It constructs features of the kind $a_i = v_{ij}$ (formulated as $a_i\_v_{ij}$). In addition to such simple features, it constructs also conjuncts (of size 2) of such features, e.g., $a_i = v_{ij} \wedge a_k = v_{kl}$, formulated as $a_i\_v_{ij}\_\_a_k\_v_{kl}$. To avoid confusion in case the same attribute name appeared in several tables, the actual form of features is $t\_a_i\_v_{ij}$ including the indicator of the name of table t in which attribute a_i appears. For a simple example of how such features are generated, the reader is referred to Appendix A.

4.2.3 Wordification

Given that in a previous experimental evaluation of propositionalization algorithms (Perovšek et al. 2013, 2015) the Wordification algorithm was shown to be the most effective, we selected Wordification as the propositionalization algorithm of choice in the proposed implementations combining propositionalization and embeddings in Sect. 6, where the Wordification algorithm was adapted to handle large data sets.

In the Wordification implementation, described in detail in Sect. 6.2.1, the original feature representation *TableName_AttributeName_AttributeValue* was—for implementational convenience—replaced by a tuple representation (*t.name, c, v*), where *t.name* refers to a table name, c to a given colon (attribute) in the table t, and v to a given value v of attribute c. Such features will be referred to as *features* or as *relational items* in the algorithm description, as appropriate.

Using this feature representation, Wordification of a multi-relational database can be summarized as the following operation:

$$\mathrm{DB}_i = \biguplus_{t \in \mathcal{T}} \mathrm{WORDIFY}(t(m(i)))$$

where m maps a given table t's indices to target (initial) table indices (i) and $\mathcal{T}$ is the set of all tables from which a foreign key path exists to the target table. The $\uplus$ operator represents a disjoint union of multisets (sum), yielding a single multiset (duplicates are allowed).

Foreign keys are designated columns that link data between distinct tables. Value of a foreign key in a given table is referred to as the instance id (the row is uniquely determined by this value). Let C represent the set of all columns that are not foreign keys, ids or target

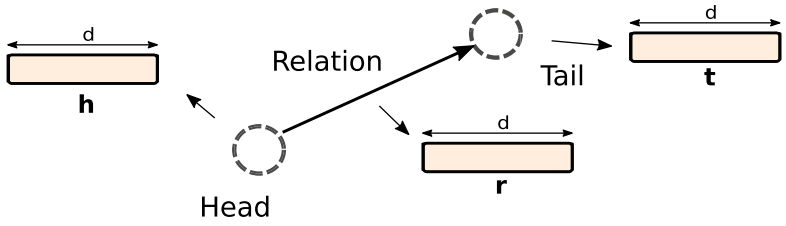


Fig. 1 Schematic representation of knowledge graph embedding. Head-Relation-Tail (h, r, t) triplets are used as inputs. Triplets are embedded in a common d-dimensional vector space

classes. The WORDIFY method returns a multiset (a bag) of relational items (for the i-th instance) constructed as follows:

$$\text{WORDIFY}(t(m(i))) = \biguplus_{v \in t[m(i)][c \in C]} (t.name, c, v)$$

where $t[c]$ represents the values v of table t in column c, and t.name is the name of table t. Thus, Wordification is naïve in the sense that it simply concatenates attribute values across tables by maintaining the column and table name information in constructing features. The original implementation, however, can become spatially intractable (see (Perovšek et al. 2013), proof of complexity) as its spatial complexity is $\mathcal{O}(\text{row} \cdot \text{tables} \cdot 2^{\text{col}})$. Details of a more efficient implementation of Wordification are available in Sect. 6.2.1.

4.3 Embedding relational structures

In this section, we discuss methodologies capable of embedding relational structures. We start with an introduction to knowledge graph embeddings, an emerging group of methods that operate on large, real-world, annotated graphs, in Sect. 4.3.1. We proceed by the presentation of entity embeddings, a more general methodology capable of supervised, as well as unsupervised embeddings of many entities, including texts and knowledge graphs in Sect. 4.3.2. Finally, in Sect. 4.3.3, we present Deep Relational Machines, an emerging methodology that links symbolic representations to deep neural networks.

4.3.1 Knowledge graph embeddings

In knowledge graphs (KG), edges correspond to relations between entities (nodes) and the graphs present Subject-Predicate-Object *triplets*. The KG handling algorithms attempt to solve the problems like triplet completion, relation extraction, and entity resolution. The KG embedding algorithms, briefly discussed below, outline some of the key ideas which render these methods highly scalable and useful for large, semantics-rich graphs. For detailed description and a recent, extensive overview of the field, we refer the reader to Wang et al. (2017), from where we next summarize some of the key ideas underlying knowledge graph embedding.

In the below description of KG embedding algorithms, the Subject-Predicate-Object triplet notation is replaced by the (h, r, t) triplet notation, where h is referred to as the *head* of a triplet, t as the *tail*, and r as the *relation* connecting the head and the tail. A schematic representation of triplet embedding is shown in Fig. 1. The embedding

methods briefly outlined below optimize the total plausibility of the input set of triplets, where plausibility of a single triplet is denoted with $f_r(h, t)$.

- The first group of KG embedding algorithms are termed *translational distance models*, as they exploit distance-based scoring functions. They measure the plausibility of a fact as the distance between the two entities, usually after a translation carried out by the relation. One of the representative methods for this type of embedding is transE (Bordes et al. 2013), where the cost function being optimized can be stated as:

$$f_r(h, t) = -||\mathbf{h} + \mathbf{r} - \mathbf{t}||^2.$$

  For vectors **h**, **r**, and **t** in the obtained embedding, score $f_r(h, t)$ is high if triplet (h, r, t) is present in the data.
- The second group of KG embedding algorithms is not deterministic, as it takes into account the uncertainty of observing a given triplet. A representative method for this type of embeddings is KG2E (He et al. 2015), which models the triplets with multivariate Gaussians. It models individual entities, as well as relations as vectors, drawn from multivariate Gaussians, assuming that **h**, **r** and **t** vectors are normally distributed, with mean vectors $\mu_h, \mu_r, \mu_t \in \mathbb{R}^d$ and covariance matrices $\Sigma_h, \Sigma_r, \Sigma_t \in \mathbb{R}^{d\times d}$, respectively. KG2E uses Kullback-Liebler divergence to directly compare the distributions as follows:

$$\begin{aligned} f_r(h, t) &= \mathrm{KL}(\mathcal{N}(\mu_t - \mu_h), \mathcal{N}(\mu_r)) \\ &= \int \mathcal{N}_x(\mu_t - \mu_h, \Sigma_t + \Sigma_h) \ln \frac{\mathcal{N}_x(\mu_t - \mu_h, \Sigma_t + \Sigma_h)}{\mathcal{N}_x(\mu_r, \Sigma_r)} dx, \end{aligned}$$

  where $\mathcal{N}_x$ denotes the probability density function of the normal distribution.
- Semantic matching models exploit similarity-based scoring functions. They measure plausibility of facts by matching latent semantics of entities and relations embodied in their vector space representations. One of the representative algorithms for learning by semantic matching is RESCAL (Nickel et al. 2011). RESCAL optimizes the following expression:

$$f_r(h, t) = \mathbf{r}^T \cdot M_r \cdot \mathbf{t},$$

  where **h** and **t** are representations of entities, and $M_r \in \mathbb{R}^{d\times d}$ is a matrix associated with relations.
- Matching using neural networks. Deep neural networks model triplets via training of neural network architectures. One of the first approaches was Semantic Matching Energy (SME) (Bordes et al. 2014). This method first projects entities and their relations to their corresponding vector embeddings. The relation's representation is next combined with the relation's head and tail entities to obtain $g_1(\mathbf{h}, \mathbf{r})$ and $g_2(\mathbf{t}, \mathbf{r})$ entity-relation representations in the hidden layer. Finally, a dot product is used to score the triplet relation matching

$$f_r(h, t) = g_1(\mathbf{h}, \mathbf{r})^T \cdot g_2(\mathbf{t}, \mathbf{r}).$$

  The simplest version of SME defines the g_1 and g_2 as:

$$g_1(\mathbf{h}, \mathbf{r}) = W_1^{(1)} \cdot \mathbf{h} + W_1^{(2)} \cdot \mathbf{r} + b_1$$
$$g_2(\mathbf{t}, \mathbf{r}) = W_2^{(1)} \cdot \mathbf{t} + W_2^{(2)} \cdot \mathbf{r} + b_2.$$

Here, $W_1^{(1)}, W_1^{(2)}, W_2^{(1)}$ and $W_2^{(2)}$ are $\mathbb{R}^{d \times d}$ dimensional weight matrices and b_1 and b_2 are bias vectors.

Recent advances in embeddings of knowledge graphs show interesting research directions. For example, hyperbolic geometry could be used to better capture latent hierarchies, commonly present in real-world graphs (Nickel and Kiela 2017). Further, KG embedding methods are increasingly tested on large, multi-topic data collections, for example, the Linked Data (LD) which standardize and fuse data from different resources. Knowledge graph embeddings, such as RDF2vec (Ristoski and Paulheim 2016) attempt to exploit vast amounts of information in LD and transform it into a learning-suitable format. As knowledge graphs are not necessarily the only source of available information, algorithms exploit also other information, e.g., textual information available for each triplet (Wang et al. 2014). Recent trends in knowledge graph embeddings also explore how symbolic, logical structures could be used during embedding construction. Approaches such as KALE (Guo et al. 2016) construct embeddings by taking into account logical rules (e.g., Horn clauses) related to the knowledge graph, thus increasing the quality of embeddings. Similar work was proposed by Rocktäschel et al. (2015), where pairs of embeddings were considered during optimization. The same group also showed how relations can be modeled without grounding the head and tail entities for simple implication-like clauses (Demeester et al. 2016). Wang et al. (2015) demonstrated that logical rules can aid in knowledge graph completion on large knowledge bases. They showed that inclusion of rules can reduce the solution space and significantly improve the inference accuracy of embedding models.

4.3.2 Entity embedding with the StarSpace approach

The guiding principle behind all embeddings, described in the previous section, is the persistence of similarity, i.e. that entities which are similar in the knowledge graph must be represented by vectors that are similar in the embedding space. A general approach implementing this principle is to use any similarity function between entities to form a prediction task for a neural network. Below we describe a successful example of this approach, called StarSpace (Wu et al. 2018). As this approach assumes discrete features from a fixed dictionary, it is particularly appealing to relational learning and inductive logic programming.

The idea of StarSpace is to form a prediction task where a neural network is trained to predict the similarity between an entity and its related entity (e.g., its label or some other entity). The resulting neural network can be used for several purposes: directly in classification, to rank instances by their similarity, or weights of the trained network can be used as pretrained embeddings.

In StarSpace, each entity has to be described by a set of discrete features from a fixed-length dictionary and forms a so called Bag-Of-Features. This representation is general enough to cover texts (documents or sentences can be described by bags-of-words or bags-of-n-grams), users (described by bags of documents, movies, or items they like), relations and links in graphs (described by semantic triples), etc. During training, entities of different kinds are embedded *in the same* latent space, suitable for various down-stream learning tasks, e.g., a user can be compared with the recommended items. Note that entities can be embedded along with target classes, resulting in *supervised embedding learning*. This type

of representation learning is the key element of the proposed PropStar algorithm outlined in Sect. 6.1.2 and presented in detail in Sect. 6.2.3.

The StarSpace approach trains a neural network model to predict which pairs of entities are similar and which are dissimilar. Two kinds of training instances are formed, positive $(a, b) \in E^+$, which are task dependent and contain correct relations between entities (e.g., document a with its correct label b), and negative instances $(a, b_1^-), \ldots, (a, b_k^-) \in E_a^-$. For each entity a (e.g., a document) appearing in the positive instances, negative instances are formed using k-negative sampling from labels $\{b_i^-\}_{i=1}^k$ as in word2vec (Mikolov et al. 2013). In each batch, the neural network tries to minimize the loss function L, defined as follows:

$$\mathrm{L} = \sum_{(a,b)\in E^+} \left(\mathrm{Loss}(\mathrm{sim}(a,b)) + \frac{1}{k} \sum_{\substack{i=1 \\ (a,b_i^-)\in E_a^-}}^{k} \mathrm{Loss}(\mathrm{sim}(a,b_i^-)) \right)$$

For each batch update in the training of neural network, k negative examples (a parameter) are formed by randomly sampling labels b_i^- from within the set of entities that can appear in b. For example, in the document classification task, document a has its correct label b, while k negative instances have their labels b_i^- sampled from the set of all possible labels. Similarity function *sim* represents the similarity between the vector representations of the two entities; typically a dot product similarity is used. Within one batch, loss function Loss sums the losses of the positive instance (a, b) and the average of the k negative instances $(a, b_i^-), i \in 1 \ldots k$. To asses the loss, margin ranking loss is used, $\mathrm{Loss} = \max(0, m - \mathrm{sim}(a, b'))$, where m is the margin parameter, i.e. the similarity threshold, and b' is a label.

The trained network can be used for several purposes. To classify a new instance a, one iterates over all possible labels b' and chooses $\arg\max_{b'} \mathrm{sim}(a, b')$ as the prediction. For ranking, entities can be sorted by their predicted similarity score. The embedding vectors can also be extracted and used for some other downstream task. Wu et al. (2018) recommend that the similarity function $sim(\cdot, \cdot)$ is shaped in such a way that it will directly fit the intended application, so that training will be more effective.

A few examples of tasks successfully tackled with the StarSpace feature transformation approach are described below.

- In *multiclass text classification* the positive instances (a, b) are taken from the training set of documents E^+, represented with bags-of-words and their labels b. For negative instances, entities b_i^- are sampled from the set of possible labels.
- In *recommender systems* users are described with a bag of items they liked (or bought). The positive instances use a single user ID as a and one of the items that user liked as b. Negative instances take b_i^- from the set of possible items. Alternatively, to work for new users, the a part of user representation is composed of all the items that user liked, except one, which is used as b.
- For *link prediction* the concepts in a graph are represented as triples head-relation-tail (h, r, t), e.g., gene-generates-protein. A positive instance a consists either of h and r, while b consists of t; alternatively, a consists of h, and b consists of r and t. Negative instances b_i^- are sampled from the set of possible concepts. The trained network can then predicted links, e.g., gene-generates-what.

Fig. 2 Example input data for a deep relational machine that operates on the instance level

Instance	$f_1 \wedge f_2$	$f_3 \wedge f_2$	$f_1 \wedge f_3$	$f_5 \wedge f_2$	$f_4 \wedge f_1 \wedge f_5$	Class
1	[1	1	1	1	0]	+
2	[0	1	0	0	1]	+
3	[0	0	1	0	0]	-
4	[0	1	0	0	1]	-
5	[1	0	0	0	1]	-

- For *sentence embedding* in an unsupervised fashion, a collection of documents, containing sentences, is turned into a training set. For positive instances, a and b are sentences from the same document (or are close together in a document), while for negative instances, sentences b_i^- are coming from different documents. This definition of a task tries to capture the semantic similarity between sentences in a document.

In the PropStar algorithm proposed in this work, we use StarSpace similarly to the first case mentioned above (multiclass text classification). Namely, Wordification returns a bag of features (relational items) for each instance in the target table. The embeddings are learned for each feature separately, and class labels are also embedded in the same space. During classification, representations of relational items associated with a given instance (bag of features) are averaged to obtain the representation of the instance—a similar idea as in the document representation adopted in the highly efficient doc2vec branch of algorithms aimed at document classification (Le and Mikolov 2014). The embedded instances, now located in the same vector space as the embeddings of class labels, are directly used for classification. The label, closest to the representation of a given target instance is selected as the final prediction.

4.3.3 Deep relational machines

Deep neural networks are effective learners in numeric space, capable of constructing intermediate knowledge constructs and thereby improve semantics of baseline input representation. Training deep neural networks on propositionalized relational data were explored by Srinivasan et al. (2019), following the work of Lodhi (2013), where Deep Relational Machines (DRMs) were first introduced. In Lodhi's work, the DRMs used bodies of first order Horn clauses as input to restricted Boltzmann machines, where conjuncts of bonds and other molecular structure information compose individual complex features; when all structural properties are present in a given instance, the target's value is true, and false otherwise. For example, consider the following propositional representation of five instances (rows), where complex features are comprised of conjuncts of atoms f_i, as illustrated in Fig. 2.

Note that the propositionalized data set P is usually a sparse matrix, which can represent additional challenge for neural networks. The DRMs proposed by Lodhi (2013) were used for prediction of protein folding properties, as well as mutagenicity assessment of small molecules. This approach used feature selection with information theoretic measures such as information gain as the sparse matrix resulting from the propositionalization was not

suitable as an input to the neural network. The initial studies regarding DRMs explored how deep neural networks could be used as an extension of relational learning.

Recently, promising results were demonstrated in the domain of molecule classification (Dash et al. 2018) using ILP learner Aleph in its propositionalization mode for feature construction. After obtaining propositional representation of data, the obtained data table was fed into a neural network that associated such representations with the output space (e.g., a molecule's activity). Again, sparsity and size of the propositionalized representation is a problem for deep neural networks. Again, stochastic feature selection of relational features that are used as input to deep relational machines can improve the performance and interpretability (Dash et al. 2019).

The work of Srinivasan et al. (2019) is relevant for the interpretability of deep relational machines, proposing a logical approximation of well-known prediction explanation method LIME (Ribeiro et al. 2016) and showing how it can be efficiently computed.

In summary, DRMs address the following issues at the intersection of deep learning and relational learning:

- DRMs demonstrated that deep learning on propositionalized relational structures is a sensible approach to relational learning.
- Their input is comprised of logical conjuncts, offering the opportunity to obtain human-understandable explanations.
- DRMs were successfully employed for classification and regression.
- Emerging ideas in the area of representation learning have only recently been explored in the ILP context (Dumančić et al. 2018), indicating there are many possible improvements both in terms of execution speed, as well as more informative feature construction on the symbolic side of computation.

We further discuss DRMs in the context of efficiency of their implementation in Sects. 6.1.1 and 6.2.2. Development of DRMs that are efficient with respect to both space and time is an ongoing research effort. Building on the ideas of DRMs, we implemented a variant of this approach, capable of learning directly from large, sparse matrices that are returned from Wordification of a given relational database, rather than using feature selection or the output of Aleph's feature construction approach. Our novel, efficient DRM implementation is presented in Sect. 6.2.2.

5 Unifying framework for propositionalization and embeddings

The connection we made between different information representation levels and different transformation techniques shows that propositionalization and embeddings are two sides of the same coin. If we view embeddings as transformations for texts, graphs, recommendations, electronic health records, and other entities with defined similarity function, we can conclude that all these transformation present a multifaceted approach to feature construction.

To this end, the paper contributes a novel understanding of these data transformation techniques. In Sect. 5.1, we first present a unified terminology and definitions, and explain the apparent differences between the definitions of propositionalizationa and embeddings as variants of a complex data transformation task. In further sections we explore the apparent differences between the two approaches. In Sects. 5.2, 5.3, and 5.4 we discuss

Table 1 Unifying and differentiating aspects of propositionalization and embeddings in terms of data representation

Representation	Propositionalization	Embeddings
Vector space	Symbolic	Numeric
Features/variables	Symbolic	Numeric
Feature values	Boolean (0 or 1)	Numeric
Sparsity	Sparse	Dense
Space complexity	Space consuming	Mostly efficient
Interpretability	Interpretable features	Non-interpretable

differences in data representation, learning, and use. Finally, in Sect. 5.5 we summarize strengths and limitations of propositionalization and embeddings.

5.1 Unifying definitions

Below we present a unified view on the definitions of propositionalization and embedding tasks, as instances of a general data transformation task defined in Sect. 1 via Definition 1.

Definition 2 (*Propositionalization*)

Given: Input data of a given data type and format, and heterogeneous background knowledge of various data types and formats.
Find: A tabular representation of the data enriched with the background knowledge, where each row represents a single data instance, and each column represents a feature in a d-dimensional symbolic[1] vector space F^d.

Definition 3 (*Embedding*)

Given: Input data of a given data type and format, and heterogeneous background knowledge of various data types and formats.
Find: A tabular representation of the data enriched with the background knowledge, where each row represents a single data instance, and each column represents one of the dimensions in the d-dimensional numeric vector space $\mathbb{R}^d$.

5.2 Unifying propositionalization and embeddings in terms of data representation

Both data transformation techniques result in a vector space representation. The unifying dimensions of propositionalization and embeddings in terms of data representation, which are summarized in Table 1, are explained below.

In propositionalization, the transformation results in a binary matrix of sparse binary vectors, where rows corresponds to training instances and columns correspond

[1] In the case of binary valued features, each value in each column is $\in \{0, 1\}$.

Table 2 Unifying and differentiating aspects of propositionalization and embeddings in terms of learning context

Learning	Propositionalization	Embeddings
Meaning capturing	Via symbols	Via distances
Search strategy	Heuristic search	Greedy
Search goal	Global optimum	Local optimum
Typical algorithms	Symbolic, linear regression, SVM	Deep neural networks
Parameters	Few	Many
Hardware	CPU	CPU/GPU

to symbolic features constructed by a particular propositionalization algorithm. These features are human interpretable, as they are either simple logical features (such as attribute values), conjunctions of such features, relations among simple features (such as e.g., a test for the equality or inequality of values of two attributes of the same type), or relations among entities (such as links among nodes in a graph). Given that the number of constructed features is usually large, such transformation results in a sparse binary matrix with few non-zero elements.

Embeddings output is usually a dense matrix of a user-defined dimensionality, composed of vectors of numeric values, one for each object of interest. For neural network based embeddings, vectors usually represent the activation of neural network nodes of one or more levels of a deep neural network. Given a relatively low dimensionality of these vectors (from 100 to 1000) this dense representation is efficient in terms of space. However, the features/dimensions are non-interpretable, therefore a separate explanation mechanisms and visualizations are required.

5.3 Unifying propositionalization and embeddings in terms of learning

For both data transformation techniques, the resulting vector space representation is used as an input to a learning algorithm of the user's choice. The unifying dimensions of propositionalization and embeddings in terms of most frequently used learners (summarized in Table 2) are explained below.

After propositionalization, any learner capable of processing symbolic features can be used. Typical learners include rule learning, decision tree learning, random forests for a supervised setting, or association rules and symbolic clustering algorithms applied in a non-supervised learning setting. Learners usually use heuristic search to find a global optimum in terms of heuristics to be optimized (exceptions being, e.g., association rule learners using exhaustive search with constraints). Typical algorithms are decision tree learners, rule learners, linear regression and SVMs. Learners require some parameter tuning to achieve optimal results, but parameters are relatively few. Learning is typically performed on CPUs.

The embedded vectors are best suited for distance-based learners, such as neural networks, and to a lesser degree for kernel methods or logistic regression. Deep neural networks use greedy search to find locally optimal solutions, and are usually trained on GPUs, but can be used for prediction on both CPUs or GPUs. As a weakness, deep learning algorithms require substantial (hyper)parameter tuning.

Table 3 Unifying and differentiating aspects of propositionalization and embeddings in terms of use

Use	Propositionalization	Embeddings
Problems/context	Relational	Tabular, texts, graphs
Data type fusion	Enabled	Enabled
Explanation	Directly interpretable	Special approaches

5.4 Unifying propositionalization and embeddings in terms of use

The unifying dimensions of propositionalization and embeddings in terms of their use (summarized in Table 3) are explained below.

Propositionalization (Kramer et al. 2001) is one of the established methodologies used in relational learning (Džeroski and Lavrač 2001; De Raedt 2008) and ILP (Muggleton 1992; Lavrač and Džeroski 1994; De Raedt 2008) (see the propositionalization methods outlined in Sect. 4.2). The propositionalization approach was applied also in the semantic data mining where ontologies are used as a background knowledge in relational learning (Podpečan et al. 2011; Lavrač et al. 2009; Vavpetič and Lavrač 2011).

The embedding technologies are mostly used in the context of deep learning for various data formats, including tabular data, texts, images, and graphs (including knowledge graphs). In addition to knowledge graph embedding approaches (see Sect. 4.3.1), we outline some other approaches to graph embeddings below.

The first studies of graph embeddings were influenced by embedding construction from textual data. For example, the well known skip-gram model, initially used as part of word2vec (Mikolov et al. 2013) was successfully applied to learn node representations. DeepWalk (Perozzi et al. 2014) was one of the first learners that treats short random walks in graphs as sentences (or short phrases) to learn latent node embeddings. DeepWalk was revisited as node2vec (Grover and Leskovec 2016) to take into account different types of random walks, parameterized by breadth, as well as depth-first search. LINE (Tang et al. 2015b) performs similarly well for the tasks of classification and link prediction by attempting to optimize both local, as well as global network structure.

As for fusing heterogeneous data types, a propositionalization approach was proposed as a mechanism for heterogeneous data fusion (Grčar et al. 2013). As for data type fusion using embedding-based methods, PTE (Tang et al. 2015a) exploits heterogeneous networks of texts for supervised embedding construction. NetMF (Qiu et al. 2018) is a generalization of Deepwalk, node2vec, LINE and PTE, re-formulating them as a matrix factorization problem. Furthermore, struc2vec (Ribeiro et al. 2017) builds on two main ideas: representations of two nodes must be close if the two nodes are structurally similar, and the latent node representation should not depend on any node or edge attribute, including the node labels. Examples of approaches to heterogeneous graph embeddings include HINMINE (Kralj et al. 2018), metapath2vec (Zhu et al. 2018) and OhmNet (Žitnik and Leskovec 2017), an extension of node2vec to a heterogeneous biological setting. Heterogeneous data embeddings (Chang et al. 2015) of images, videos and text were also formulated as a task of heterogeneous graph embedding.

Concerning the interpretability of results, propositionalization approaches are mostly used with symbolic learners whose results can be interpretable, given the interpretability of features used in the transformed data description. For embedding-based methods, given the non-interpretable numeric features/dimensions, specific mechanisms need to be implemented to ensure results explanation (Robnik-Šikonja and Kononenko 2008; Štrumbelj and

Kononenko 2014). A recent well-known approach, which can be used in a post-processing phase of an arbitrary prediction model, is named SHAP (Lundberg and Lee 2017). In this approach, Shapley values offer insights into instance-level predictions by assigning fair credit to individual features for participation in prediction-explaining interactions. Explanation methods such as SHAP are commonly used to understand and debug black-box models. We refer the reader to Lundberg and Lee (2017) for a detailed overview of the method.

5.5 Summary of strengths and limitations of propositionalization and embeddings

Let us summarize the unified presentation of propositionalization and embeddings by presenting the strengths and weaknesses of the two approaches. The main strength of propositionalization is the interpretability of the constructed features, while the main strength of embeddings is high performance of classifiers learned from embeddings due to their compact representation in a vector space.

In terms of their strengths, both approaches to data transformation are: (a) automated, (b) fast, (c) semantic similarity of instances is preserved in the transformed instance space (as a remark, due to a more compact representation, embeddings preserve semantic similarity of features even better than propositionalization), (d) transformed data can be used as input to standard propositional learners, as well as to contemporary approaches.

In addition to these characteristics, embeddings have other favorable properties: (a) embedded vectors representations allow for transfer learning, e.g., for cross-lingual applications in text mining or image classification from different types of images, (b) cover a very wide range of data types (text, relations, graphs, images, time series), and (c) have a very wide community of developers and users, including industry.

In terms of their limitations when used in a multi-relational setting, both approaches to data transformation: (a) are limited to 1-many relationships (cannot handle many-to-many relationships between the connected data tables), (b) cannot handle recursion, and (c) cannot be used for predicate invention.

In addition to these characteristics, limitations of propositionalization include: (a) only boolean values are used in the transformed vector space, (b) generated sparse vectors can be memory inefficient, (c) limited range of data types are handled (relations, graphs), and (d) a small community of developers and users (mainly from ILP).

Embeddings also have several limitations: (a) loss of explainability of features and consequently of the models trained on the embedded representations, (b) many user-defined hyper-parameters, (c) high memory consumption due to many weights in neural networks, and (d) requirement for specialized hardware (GPU) for efficient training of embeddings, which may be out of reach for many researchers.

6 Proposed unification methodology and its two implementations

The unifying aspects analyzed in Sect. 5 can be used as a basis for a unifying methodology that combines propositionalization and embeddings, and benefits from the advantages of both. The propositionalization successfully captures relational information through complex relational feature construction, but results in a sparse symbolic feature vector representation. This weakness can be successfully overcome by embedding the constructed feature

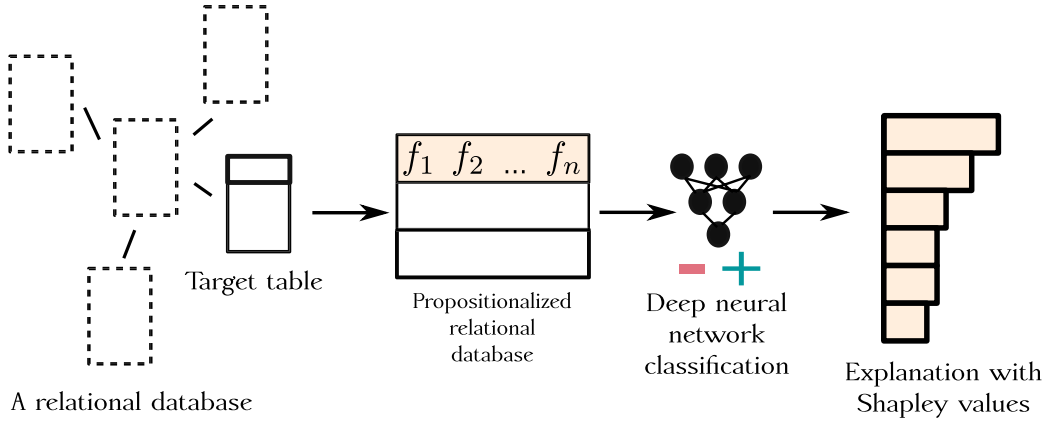


Fig. 3 Overview of the PropDRM instance-based embedding methodology, based on DRMs. Note that features in the propositionalized relational database represent either single features f_i or conjuncts of features, e.g., $f_i \wedge f_j$, given that Wordifications constructs both feature forms. For simplicity, the propositionalized database shows only two instances

vectors into a lower dimensional numeric vector space, resulting in a condensed numeric feature vector representation appropriate for use by modern deep learning algorithms.

To this end, we describe two novel data transformation algorithms, combining propositionalization and embedding based transformations into a joint data transformation framework. We first briefly outline the two approaches in Sect. 6.1, followed by their detailed descriptions in Sect. 6.2.

6.1 Outline of proposed data transformation and learning methods

We first overview the proposed unifying data transformation approaches. The first, named PropDRM, is an instance-based data transformation approach. The second one is a feature-based data transformation pipeline, called PropStar. The approaches are outlined in the next two subsections.

6.1.1 PropDRM: an instance-based approach

The first unifying approach for embedding of multi-relational databases is based on Deep Relational Machines (Dash et al. 2018) (DRMs), presented in Sect. 4.3.3. Rather than using the output of Aleph's feature construction approach, as was the case in the DRM implementation of Dash et al. (2018), we implemented a variant of this approach, capable of learning directly from large, sparse matrices that are returned by the Wordification (Perovšek et al. 2015) approach to propositionalization of relational databases. In this work, following the paradigm of propositionalization by Wordification, each instance is described by a bag (a multiset that allows for multiple appearances of its elements) of features of the form *TableName_AttributeName_Value*. Wordification treats these simple easily interpretable features as 'words' in the transformed Bag-Of-Words representation. In this work, they represent individual 'relational items' and we use the notation (table.name, column.name, value).

Relational representations are thus obtained for individual instances, resulting in embeddings of instances (e.g., molecules, persons, companies etc). Batches of instances are then

fed to a neural network, which performs the desired down-stream task, such as classification or regression. Schematically, the approach is illustrated in Fig. 3.[2]

Note that although propositionalization and subsequent learning are conceptually two distinct steps, they are not necessarily separated when implemented in practice: as neural networks operate with small batches of input data, if propositionalization is capable of similar batch functionality, relational features can be generated in a lazy manner when needed by the neural network. The technical details of the proposed PropDRM implementation are presented in Sect. 6.2.2.

When compared to our PropStar algorithm presented in Sects. 6.1.2 and 6.2.3 below, the key difference of the outlined DRM-based implementation of the unifying methodology is the type of embeddings: PropDRM embeds instances (i.e. whole bags of constructed features), whereas PropStar embeds features along with the class values in the same vector space.

6.1.2 PropStar: a feature-based approach

In this section, we outline the proposed PropStar algorithm for classification via feature embedding. Its details and implementation are presented in Sect. 6.2.3. Unlike the PropDRM algorithm, where each embedding vector represents a single data *instance*, the idea of PropStar is to use embedding vectors to represent the *features* of the data set. Here, individual relational features, obtained as the result of propositionalization by Wordification, are used by a supervised embeddings learner to obtain representations, co-located with instance labels. This approach is conceptually different in the sense that representations are not learned for individual instances (as is the case of DRMs); instead, they are learned for every single relational feature that is the output of the selected propositionalization algorithm (i.e. Wordification).

The fact that PropStar produces vector representations of features means that the labels (label=true and label=false) are also represented by vectors in the same dense space as the other vectors. This leads to an intuitive direct classification of new examples. We can observe the set of vectors representing the relational items present in the itemset representing the new example. To classify a new instance, the embeddings of the set of its features (i.e. true values) are averaged and the result is compared to the embedding of class labels. The nearest class label is chosen as the predicted value.

Figure 4 illustrates how new instances are classified by direct comparison of the representations of their features in the latent dense semantics-preserving space that also contains the information on labels. The classification is based on the proximity to a given label (in the latent space). If the center of feature vectors of a given instance is closer to the vector representing the feature label=true, then the example is classified as positive.

In contrast to the instance-based embeddings discussed in Sect. 6.1.1, which relies on batches, the whole data set is needed to obtain representations for individual features. To avoid high spatial complexity, this class of algorithms would ideally operate on sparse inputs. An example of feature-based embeddings are items that are to be recommended to users, where the representation of a given item is obtained by jointly optimizing the item's co-occurrence with other items, as well as other user's properties. In

[2] As its last step, the methodology includes the explanation of results using the SHAP approach. However, as Sect. 6 focuses on our research contributions, this well known approach and its results are presented in Appendix D.

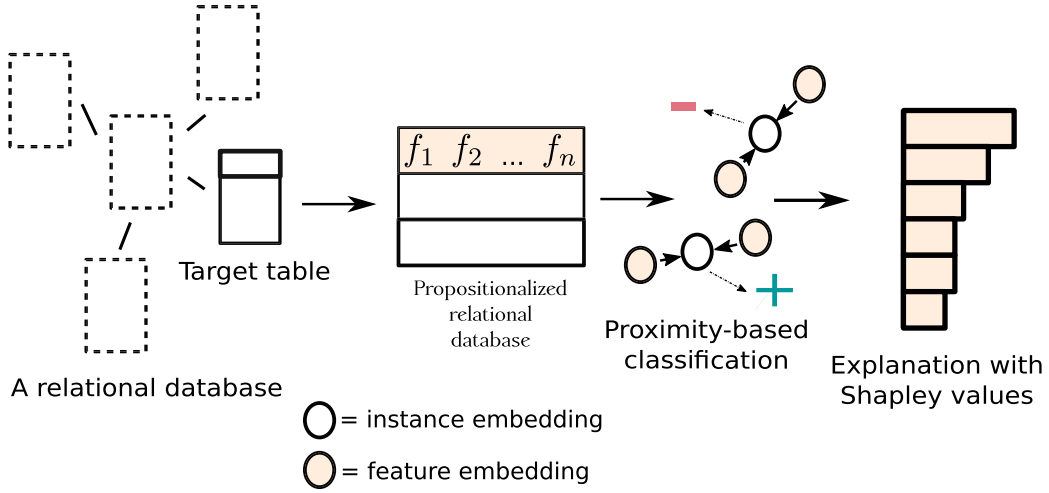


Fig. 4 Overview of the proposed feature-based embedding methodology PropStar. Note that embedded features represent embeddings of single features f_i or of conjuncts of features, e.g., $f_i \wedge f_j$, given that Wordifications constructs both feature forms. For simplicity, the propositionalized database shows two instances Blank and shaded circles correspond to embedded representations of instances and features, respectively

a relational setting considered in this work, we follow the paradigm of propositionalization by Wordification, where each instance is described by a bag of features of the form (table.name, column.name, value). Consequently, in the PropStar approach the embeddings represent bags of such features and their conjunctions (of size 2). There are as many embeddings as there are unique *features* in the propositionalized representation of a given relational database. As such embeddings by themselves do not contain any information which relates them to the desired output space, target values get embedded alongside other features in a supervised manner.

6.2 Detailed description of proposed data transformation and learning methods

This section presents the implementations of the proposed methods, preceded by the description of the updates to the Wordification algorithm (Perovšek et al. 2015 for multi-propositionalization algorithm presented in Sect. 6.2.1. In Sect. 6.2.2 we discuss how Deep Relational Machines (described briefly in Sect. 4.3.3), which use neural networks for learning from relational databases, were adapted to operate on sparse matrices generated by an improved Wordification algorithm. In Sect. 6.2.3 we describe a novel algorithm, called PropStar, which embeds relational features, extracted as part of propositionalization.

6.2.1 Improving the efficiency of Wordification

In this work we significantly extend the ideas proposed in Wordification (Perovšek et al. 2013, 2015) with the aim to maintain the classification performance, yet improve its *scalability*. Both proposed algorithms build on the idea of Wordification, yet its use in our algorithms is differentiated by the following design decisions:

1. Inputs do not need to be hosted in relational databases. PropStar operates on .sql files directly. The algorithm supports SQL conventions, as commonly used in the ILP com-

munity.[3] This modification renders the method completely local, enabling offline execution without additional overhead. Such setting also offers easier parallelism across computing clusters.
2. Algorithm is implemented in Python 3 with minimum dependencies for computationally more intense parts, such as the Scikit-learn (Pedregosa et al. 2011), Pandas, and Numpy libraries (Van Der Walt et al. 2011). All database operations are implemented as array queries, filters or similar, unlocking the potential to run PropDRM and PropStar also on GPUs.
3. As shown by Perovšek et al. (2015), Wordification's caveat is extensive sampling of (all) tables. We relax this constraint to close (up to second order) foreign key neighborhood, notably speeding up the relational item sampling part, but with some loss in terms of relational item diversity. For larger databases, minimum relational item frequency can be specified, constraining potentially noisy parts of the feature space.

One of the original Wordification's most apparent problems is its spatial complexity. In this work we address this issue as follows:

1. Relational items are hashed for minimal spatial overhead during sampling.
2. During construction of the final representation, a sparse matrix is filled based on relational item occurrence.
3. The matrix is serialized directly into list-like structures, suitable for StarSpace algorithm and thus we maintain minimal spatial overhead.
4. Only the final representation is stored as a low-dimensional (e.g., 32) dense matrix.

6.2.2 Detailed description of the proposed PropDRM implementation

The novelty of the proposed implementation of DRM instance-based embedding, inspired by the work of Dash et al. (2018), concerns its capability to effectively handle the sparseness of the data with deep neural networks. The main novelty of the proposed implementation is that it is indeed capable of operating on larger, sparse matrices directly. Such capability is necessary for DRMs to be compatible with propositionalization, which yields large sparse matrices as the main output. Below we discuss the neural network architecture and its adaptations.

Let P represent a sparse item matrix, as returned by Wordification (discussed in Sects. 4.2.3 and 6.2.1). Note that Wordification is unsupervised, and thus does not include any information on instance labels. The neural network we use (termed ω) represents the mapping $\omega : P \rightarrow \mathcal{C}$, where $\mathcal{C}$ is the set of classes. In this work, we experimented with dense feed-forward neural networks, regularized using dropout (Srivastava et al. 2014), and ELU activation function (Clevert et al. 2016) (of intermediary weights). The output weights are activated using sigmoid activation (σ) in order to obtain binary predictions.

$$ELU(x) = \begin{cases} c(e^x - 1), & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases},$$

where c is the user-specified constant. For a given input matrix P, an example of a single hidden-layer neural network is defined as follows.

[3] https://relational.fit.cvut.cz/.

$$\omega = \sigma(W_o^T(\text{ELU}(\text{Drop}(W_1^T P + b_1))) + b_o).$$

Here, the σ is a sigmoid activation, defined as $\sigma(x) = \frac{1}{1+e^{-x}}$. The W_1 is the weight matrix, P the sparse input space, and b_l the bias vector of a given layer $l \in \{0, 1\}$. The described neural network returned satisfactory results, hence, we did not perform neuroevolution or similar large-scale search for potentially better performing architectures. Throughout this work, we use the binary cross-entropy loss, referred to as *Loss*. For a given probabilistic classifier, which returns a probability p_{ij} of an instance i belonging to a class j, the loss function is defined as follows:

$$\text{Loss}(i) = \sum_{j \in \mathcal{C}} y_{ij} \cdot \log p_{ij}.$$

Here y_{ij} is a binary value (0 or 1) indicating whether class j is the correct class label assigned to instance i, and $\mathcal{C}$ is a set of all the target classes. In the case of DRMs, where the instances of a relational database (one of the tables) are classified, each of the $|\mathcal{C}|$ output neurons predicts a single probability p_{ij} for a given target class $j \in \mathcal{C}$. If the neural networks are trained in small batches, the results of the Loss function are averaged to obtain the overall loss of a given batch of instances.

Neural networks are adapted for dense inputs such as images and texts, and are not necessarily suitable for large sparse matrices, as considered in this work (i.e. P). The proposed variant of DRMs is adapted as follows. Once the batch size bs (a free parameter) is determined, propositionalized representation P is traversed (in chunks of bs instances). Note that each instance is effectively a d-dimensional vector. As the neural network operates with dense batches, each batch is converted to a dense matrix of $bs \cdot d$ elements that is used during matrix multiplication within the neural network. The spatial complexity is thus at most $\mathcal{O}(bs \cdot d)$. We observed that even by considering batch size of one, the DRMs are stable and efficient.

6.2.3 Detailed description of the PropStar algorithm

We next present the novel feature-based embedding algorithm that can operate directly on the propositionalized relational databases. The proposed PropStar algorithm merges symbolic and non-symbolic representations as part of a single procedure for obtaining real-valued representations of features in arbitrary relational databases. The pseudocode of the PropStar algorithm is given in Algorithm 1.

Algorithm 1: The pseudocode of PropStar algorithm.

```
Data: A Relational database R, foreign key map m
Parameters : Entity embedding parameter set E, representation dimension d,
             target table T, target attribute t
itemContainer ← empty bag of items ;                          ▷ Begin Wordification.
foreach instance i ∈ T do
    relationalItems ← {} ;
    candidateKeys ← getForeignKeys(R,i) ;                     ▷ Links to other tables.
    candidateTables ← getCandidateTables(candidateKeys, R) ;  ▷ Linked tables.
    foreach table ∈ candidateTables do
        while not final number of items do
            bagOfItems ← WORDIFY(table(m(instance))) ;
            add bagOfItems to relationalItems ;               ▷ Store sampled items.
        end
    end
    itemContainer[i].add(relationalItems) ;                   ▷ Store relational items.
end
relevantFeatures ← frequencySelection(itemContainer.values, d) ;
symRep← [] ;                                  ▷ Sparse vector representations of instances.
foreach instance i ∈ targetTable do
    instanceItems ← itemContainer[i] ;
    propRep ← RelationalFeatures(relevantFeatures, instanceItems) ;
    symRep.append(propRep) ;
end
featureEmbeddings ← StarSpace(symRep, T[t], E) ;              ▷ Input is a sparse matrix.
return featureEmbeddings ;
```

The algorithm consists of two main steps. First, a relational database is transformed into sets of features describing individual instances. The WORDIFY method constructs features of the form (table.name, column.name, value) and uses them to describe each individual instance (see Sect. 6.2.1 for a detailed formulation of this step).

Second, sets of relational items (features) are used as input to the StarSpace entity embedding algorithm (described in Sect. 4.3.2), producing embeddings for each distinct relational item. The StarSpace embeddings are computed using efficient C++ implementation. We wrote a wrapper which seemingly integrates the first part of PropStar (sampling and propositionalization) with the embedding construction. The problem is formulated as a multiclass classification, where the positive pair generator comes directly from a training set of labeled data specifying $(a, b) \in E^+$ pairs where a are relational item 'documents' and b are labels (singleton features). Negative entities b_i^- are sampled from the set of possible labels. Inputs can be described as (multi) sets comprised of both relational items f_i, their conjuncts, as well as class labels c_i. For example,

$$\{f_1, f_2, f_6, f_6 \wedge f_2, c_1\}$$

represents a simple input consisting of three relational items, a conjunct and the target label c_1. Note that we apply StarSpace in such manner that the representations are learned for *individual relational items*. A representation matrix of dimension $\mathbb{R}^{|W| \times d}$ is produced as the final output ($|W|$ represents the number of unique relational items considered). Intuitively, the embedding construction can be understood as determining relational item locations in a latent space based on their co-occurrence with other items present in all training instances. The wrapper can be called via 'fit' and 'predict' methods, commonly used in contemporary data science and machine learning. In this work, we consider the inner product similarity

between a pair of vectors $\boldsymbol{e}_1, \boldsymbol{e}_2$ for the construction of embeddings.[4], i.e. The complexity of obtainingsim$(\boldsymbol{e}_1, \boldsymbol{e}_2) = \boldsymbol{e}_1^T \cdot \boldsymbol{e}_2$. As the class labels are embedded in *the same* space as individual relational items, classification of novel bags of relational items is possible by direct comparison, as common tasks operating on word embeddings. We discuss this classification below.

Let M represent a novel instance to be classified. Note that M (without additional index) is considered a multiset of relational items. For prediction purposes, StarSpace averages the representations of relational items present in a given input instance (a bag). The representation is normalized (as during training) and compared to label embeddings in the common space. Representation of a relational bag $\boldsymbol{e}_M$ is computed (with considered hyperparameters) as:

$$\boldsymbol{e}_M = \frac{\bigoplus_{f_i \in M} \boldsymbol{e}_{f_i}}{\sqrt{|M_{\text{unique}}|}},$$

which is a d-dimensional, real-valued vector. Note that $\oplus$ in this expression denotes element-wise summation. The M_{unique} represents the set of all (unique) relational features currently considered. Note that original bags of features can be redundant (multisets), yet representations are learned for unique features. Next, the similarity of this vector is compared to the label embeddings in the same space. The label that is the most similar to e_M is the top-ranked prediction, the second most similar label is the second-ranked prediction, etc. In this work we consider only the top-ranked prediction, resulting in the following label assignment:

$$\text{label}(e_M) = \arg\max_{c \in \mathcal{C}} \left[\text{sim}(\boldsymbol{e}_M, \boldsymbol{e}_c)\right].$$

The complexity of obtaining a single prediction is hence $\mathcal{O}(|\mathcal{C}|)$, not taking the complexity of scalar product for function sim into account. The PropStar algorithm first samples the relational items with respect to the target table (lines 2-11 in Algorithm 1). Binary indicator function (relationalFeatures) is applied to obtain the propositionalized representation of the target table (line 12). Here, zeros represent absence of a given relational items, and ones their presence.[5] Finally, StarSpace is used to embed the table into a low-dimensional, real valued embedding (line 19).

The spatial complexity of PropStar is linear with respect to the number of non-zero elements in the propositionalized version of a relational database. The exact spatial complexity can be formulated as follows. Let row represent the average number of rows per table. Let n_t represent the number of tables and col the average number of columns per table. We improve the original spatial complexity of $\mathcal{O}(\text{rows} \cdot n_t \cdot 2^{\text{col}})$ by introducing a constraint, which determines the maximum number of relational items that can be considered. The exponential term in the initial complexity thus reduces to col times some constant, yielding the complexity of $\mathcal{O}(\text{rows} \cdot \text{col} \cdot n_t)$. This formulation yields a scalable propositionalization.

[4] Note that e_1, e_2 represent vector representations of relational items (i.e. features) in the output of propositionalization.

[5] Note that in the actual implementation CSR format of sparse matrices is used to reduce the spatial overhead of storing zeros.

7 Experimental evaluation

In this section we describe the implementation details of the proposed methods, the relational data sets used in the experiments, and the experimental evaluation of the proposed methods.

7.1 Implementation and hyperparameters

We discuss how the proposed methods were implemented, along with the hyperparameters explored. Both new methods (PropDRM and PropStar) are implemented in Python, with the following exceptions. In PropDRM, the DRMs are implemented in PyTorch. For PropStar we used the efficient StarSpace implementation written in C++, for which we build a wrapper offering basic embedding training and prediction functionality.

We used 10-fold stratified cross validation, which was conducted for individual hyperparameter settings. The best setting is reported, other are discussed in ablation studies. Experiments were performed on an of-the-shelf workstation with no GPUs (even though PropDRM and PropStar can exploit them). We intentionally omit the GPU-based training to explore the minimum hardware, required to perform competitively on the selected data sets—ILP baselines, such as Aleph and RSD are Prolog-based, and are to our knowledge not able to use multiple GPU threads simultaneously. The machine on which experiments were conducted had 128GB of RAM and 12 CPUs (Intel i8 series).

In PropDRM, we varied the dropout rate, learning rate, number of epochs, and the hidden layer size. In PropStar, we varied the number of negative samples, embedding dimension, learning rate, and the number of epochs.

The source code of our implementation is publicly available[6].

7.2 Relational data sets

Five relational database sources[7] (Motl and Schulte 2015) were used in the experiments. Their characteristics are summarized in Table 4.

Trains (Michie et al. 1994) data set is used in the East-West trains challenge problem, which is well-known in ILP. The East-West trains challenge is to predict whether a train is eastbound or westbound, based on the properties of eastbound and westbound cars. Trains contain variable number of cars, each having one of various shapes and carrying various loads.

Carcinogenesis (Srinivasan et al. 1997) task is to predict carcinogenicity of a diverse set of chemical compounds. The data set was obtained by testing different chemicals on rodents, where each trial would take several years and hundreds of animals. The data set consists of 329 compounds, of which 182 are carcinogens.

Mutagenesis (Debnath et al. 1991) task addresses the problem of predicting mutagenicity of aromatic and heteroaromatic nitro compounds. Predicting mutagenicity is an important task as it is very relevant to the prediction of carcinogenesis. The compounds from the data are known to be more structurally heterogeneous than in any other ILP

[6] https://github.com/SkBlaz/PropStar.

[7] Freely accessible at https://relational.fit.cvut.cz/.

Table 4 Properties of the experimental data tables

Trains	#rows	#attributes
Cars	63	10
trains	20	2
Carcinogenesis	#rows	#attributes
atom	9,064	5
canc	329	2
sbond_1	13,562	4
sbond_2	926	4
sbond_3	12	4
sbond_7	4,134	4
Mutagenesis 42	#rows	#attributes
atoms	1,001	5
bonds	1,066	5
drugs	42	7
ring_atom	1,785	3
ring_strucs	279	3
rings	259	2
Mutagenesis 188	#rows	#attributes
atoms	4,893	5
bonds	5,243	5
drugs	188	7
ring_atom	9,330	3
ring_strucs	1,433	3
rings	1,317	2
IMDB	#rows	#attributes
actors	7,118	4
directors	130	3
directors_genres	1,123	4
movies	166	4
movies_directors	180	3
movies_genres	408	3
roles	7,738	4
MovieLens	#rows	#attributes
actors	99,129	3
directors	2,201	3
movies	3,832	5
movies2actors	152,532	3
movies2directors	4,141	3
u2base	946,828	3
users	6,039	4

data set of chemical structures. The database contains 230 compounds of which 138 have positive levels of mutagenicity and are labeled as 'active'. Others have class value 'inactive' and are considered to be negative examples. We took the data sets from the original paper (Debnath et al. 1991), where the data was split into two subsets: a 188 compound data set and a smaller data set with 42 compounds.

IMDB database is publicly available in the SQL format.[8] This database contains tables of movies, actors, movie genres, directors, and director genres. The data set used in our experiments encompasses only movies whose titles and years of production appear in the IMDB's top-250 and bottom-100 chart (Snapshot taken on July 2, 2012). The snapshot contains 166 movies, along with all of their actors, genres and directors. We designate movies present in the IMDB top-250 chart as positive examples, and those in the bottom-100 as negatives.

MovieLens data set from the UC Irvine machine learning repository.[9] The data set is similar to IMDB above, however is much larger. Overall, the database consists of more than 1.2 million instances. The task is to predict gender of the movie database's users.

7.3 Results

We present the results of the empirical evaluation of the proposed methodologies on the presented set of standard benchmark ILP data sets. The accuracies of individual learners are given in Table 5, and the AUC scores are reported in Table 6. The results for Aleph, RSD, RelF and Wordification were taken from previous work of Perovšek et al. (2015).

It can be observed that the proposed unifying approaches perform competitively on most data sets. We can observe a distinct difference in performance on the Mutagenesis data sets, where both PropDRM as well as PropStar do not outperform the baselines on the smaller data set (Mut42), yet notably outperform the (best) baselines on the larger one (Mut188). Further, minor improvement over the baseline algorithms is also achieved on Carcinogenesis data set.

In terms of spatial complexity, the proposed methodology greatly outperforms the alternatives under a given set of constraints. Only PropDRM and PropStar scale to very large relational databases without specialized hardware. Detailed studies regarding the sensitivity of PropDRM and PropStar to their parameters are discussed in Appendices B and C, respectively.

We consider the presented results as very favorable for the two proposed approaches. In particular, PropStar is better than current state-of-the-art methods on 3 out of 6 data sets, and is therefore a method to take into consideration when attempting to solve any new relational problem.

7.3.1 Study of propositionalization projections

The considered propositionalization is entirely *unsupervised*. Only once the symbolic representations of instances are obtained, PropDRM and PropStar learn the associations to

[8] http://www.webstepbook.com/supplements/databases/imdb.sql.

[9] https://relational.fit.cvut.cz/dataset/MovieLens.

Table 5 Classification accuracy on different relational data sets

Propositionalization	Learner	Carc.	IMDB	Mut188	Mut42	Trains	MovieLens
	MajorityVote	0.55	0.73	0.67	0.69	0.50	0.72
Aleph (Perovšek et al. 2015)	J48	0.55	0.73	0.60	0.69	0.55	–
Aleph (Perovšek et al. 2015)	SVM	0.55	0.73	0.60	0.69	0.70	–
RSD (Perovšek et al. 2015)	J48	0.60	0.75	0.68	**0.98**	0.60	–
RSD (Perovšek et al. 2015)	SVM	0.56	0.73	0.71	0.69	0.80	–
RelF (Perovšek et al. 2015)	J48	0.60	0.70	0.75	0.76	0.65	–
RelF (Perovšek et al. 2015)	SVM	0.56	0.73	0.69	0.76	0.80	–
Wordification (Perovšek et al. 2015)	J48	0.62	0.82	0.67	**0.98**	0.50	–
Wordification (Perovšek et al. 2015)	SVM	0.61	0.73	0.82	0.79	0.50	–
Aleph (replicated)	J48	0.55	–	0.80	0.76	0.70	–
Aleph (replicated)	SVM	0.55	–	0.80	0.79	0.60	–
RSD (replicated)	J48	0.56	0.84	0.88	0.92	0.60	–
RSD (replicated)	SVM	0.60	0.82	0.89	0.84	0.80	–
Wordification (replicated)	J48	0.47	**0.85**	0.91	0.88	**0.90**	0.60
Wordification (replicated)	SVM	0.39	0.80	0.83	0.33	0.50	0.72
Treeliker	J48	0.58	–	0.77	0.81	0.50	–
Treeliker	SVM	0.60	–	0.90	0.80	0.70	–
PropDRM		0.63	0.73	0.91	0.86	0.70	0.72
PropStar		**0.66**	0.74	**0.92**	0.90	0.80	**0.74**

The best score for each dataset is in bold

For the proposed methods, we report average performance over 5 runs. The runs, marked with—were unable to finish in 12 h

individual classes. A good representation, however, already contains relevant information on the instance space. In Fig. 5, we projected the propositionalized Mutagenesis 188 and Trains instance space to two dimensions to qualitatively explore whether instances group or any meaningful patterns emerge. Understanding whether the symbolic space exhibits distinct structure on its own could offer insights into why the proposed methods perform well. For projecting the 10,000 dimensional space to two dimensions we used UMAP, a recently introduced non-linear dimensionality reduction method based on insights from manifold theory (McInnes et al. 2018).

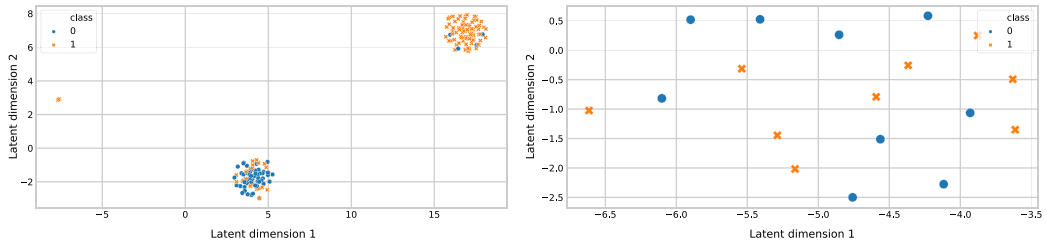
We can observe an apparent distinction in the clustering of the UMAP projections of the two propositionalized data sets. The Mutagenesis 188 data set consists of two distinct clusters that, when colored according to the class labels, approximately correspond to the two classes (Fig. 5a). On the other hand, the clustering is not apparent in the case of the Trains data set (Fig. 5b), where the instances do not group distinctly. The purpose of the considered visualizations is twofold. First, we show how the symbolic space can exhibit clustering properties, related to properties of instances such as class labels. Next, we show that projections do not necessarily exhibit such properties, indicating potentially harder classification problems. We believe that UMAP and similar tools offer insights into representation structure.

Table 6 AUC scores on individual data sets

Propositionalization	Learner	Carc.	IMDB	Mut188	Mut42	Trains	Movies
Aleph (from Perovšek et al. 2015)	J48	0.50	0.50	0.68	0.50	0.55	–
Aleph (from Perovšek et al. 2015)	SVM	0.50	0.50	0.68	0.50	0.70	–
RSD (from Perovšek et al. 2015)	J48	0.59	0.59	0.54	**0.96**	0.60	–
RSD (from Perovšek et al. 2015)	SVM	0.52	0.50	0.58	0.50	0.80	–
RelF (from Perovšek et al. 2015)	J48	0.59	0.66	0.68	0.68	0.75	–
RelF (from Perovšek et al. 2015)	SVM	0.52	0.50	0.54	0.62	0.75	–
Wordification (from Perovšek et al. 2015)	J48	0.61	**0.75**	0.55	0.96	0.95	–
Wordification (from Perovšek et al. 2015)	SVM	0.58	0.50	0.78	0.65	0.50	–
Alpeh (replicated)	J48	0.50	–	0.71	0.72	0.70	–
Aleph (replicated)	SVM	0.50	–	0.75	0.73	0.60	–
RSD (replicated)	J48	0.55	0.71	0.87	0.92	0.60	–
RSD (replicated)	SVM	0.58	0.65	0.90	0.73	0.80	–
Wordification (replicated)	J48	0.48	0.72	**0.90**	0.86	0.90	0.52
Wordification (replicated)	SVM	0.42	0.62	0.81	0.50	0.50	0.50
Treeliker	J48	0.58	–	0.75	0.71	0.50	–
Treeliker	SVM	0.58	–	0.88	0.68	0.70	–
PropDRM		**0.63**	0.68	**0.90**	0.87	0.80	0.54
PropStar		**0.63**	0.66	0.87	0.87	**0.95**	**0.56**

The best score for each dataset is in bold

We report average performance over 5 runs. The runs, marked with—were unable to finish in 12 h



(a) Mutagenesis 188. **(b)** Trains.

Fig. 5 Two UMAP projections of selected propositionalized data sets

7.3.2 Statistical comparison of PropDRM and PropStar

In previous sections, we demonstrated that both PropDRM and PropStar perform well on the considered data sets, indicating that both approaches are successfully unifying propositionalization and embeddings. We further study the differences in performances of the two approaches. For this purpose, we employ the hierarchical Bayesian t-test, a Bayesian test capable of comparing a pair of classifiers across multiple data sets (Benavoli et al. 2017; Corani et al. 2017). For this comparison, we selected the overall best performing hyperparameter sets for each method, and conducted ten repetitions of stratified ten-fold cross validation (for each data set). The results are visualized as probability distributions across the space of both classifiers and the 'rope' region (region of practical equivalence) within which the two classifiers perform the same. The size of this region is a free parameter of the hierarchical

t-test, and was set to 0.05 in this work. Other parameters of the test were left as defaults. The exact methodology for the interested reader is explained by Benavoli et al. (2017).

In terms of AUC, the probabilities returned by the Bayesian test were as follows: p(PropStar) = 0.07 and p(PropDRM) = 0.54), and in terms of classification accuracy, p(PropStar) = 0.96 and p(PropDRM) = 0.04. The results of statistical analysis indicate that with respect to AUC performance, the two approaches perform similarly, even though the probability that PropDRM will outperform PropStar is higher. With respect to the classification accuracy, PropStar outperforms PropDRM in majority of comparisons. Thus, considering the 95% or higher as the probability denoting significance boundary, we can determine that PropStar is (significantly) more suitable choice if accuracy is being optimized for. As Bayesian comparisons are computationally expensive, we compared the two methods using default hyperparameter settings. The PropStar's default configuration is not necessarily optimal when AUC is considered.

8 Conclusions and further work

This paper first provides a critical survey of propositionalization and embedding techniques, especially relevant for relational learning and inductive learning programming. While both data approaches, propositionalization and embeddings, aim at transforming data into the tabular data format, the research papers describing the approaches use different terminology and task definitions, claim to have different goals, and are used in very different contexts. In this paper, we define the main categories of data transformation techniques based on the representation they use and approaches employed. Propositionalization approaches produce tabular data from multirelational databases as well as from a mixture of tabular data and background knowledge in the form of logic programs or networked data, including ontologies. Knowledge stored in graphs can be assessed with community detection and graph traversal methods. Relations described with similarity matrices are encoded in a numeric form using matrix factorization. Currently, the most promising approach to data transformations are neural networks based methods which can be applied to text, graphs, and other entities for which we can define a suitable similarity function.

One of the main strategic problems machine learning has to solve is better integration of knowledge and models across different domains and representations. While the research area of embeddings can unify different representations in a numeric space, symbolic learning may be an essential ingredient for integration of different knowledge areas. We see our PropStar approach that combines advantages of propositionalization and neural embeddings in the same data fusion pipeline as a step in that direction.

The first minor contribution of the paper is that our exposition is based on three cognitive representation levels introduced by Gärdenfors (2000), i.e. neural, spatial, and symbolic. As most of human knowledge is stored in the symbolic form, while the most powerful machine learning algorithms take as input spatial representations, this explains a plethora of techniques that transform other forms of human knowledge into the spatial representation space. The next contribution is the unifying framework in which we describe propositionalization and embedding techniques in terms of their joint properties and their differences. We show how the propositionalization techniques can be merged with deep neural network based embedding to produce a joint embedding, such that spatial representation can be used with any deep learning algorithm and the predictions can be comprehensively explained. The main contributions of our work are thus the two implementations that merge propositionalization and embeddings in the same unifying methodology. The first is an efficient reimplementation

of existing Deep Relational Machines, while the second one is the novel Deep Propositionalization algorithm. We also contribute an experimental evaluation of the two algorithms and show favorable results in terms of predictive performance, as well as time and space requirements. The source code of both algorithms, DeepProp and PropDRM, is publicly available.[10]

In further work, it is worth investigating the integration of symbolic and deep learning, considering them as multitask learning approaches which try to integrate many different learning tasks and use embeddings as input representations. The issue is that different embedding methods have so far only been used in isolation. We already address this challenge in the current work of the authors, where we combine complementary embedding methods from different classes: in particular, to use network traversal methods to produce initial embeddings that are then refined using a deep neural network (Škrlj et al. 2019).

Acknowledgements We acknowledge the financial support of the Slovenian Research Agency through core research programmes P2-0103 and P6-0411 and project *Semantic Data Mining for Linked Open Data* (financed under the ERC Complementary Scheme, N2-0078). The authors have received funding also from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825153 (EMBEDDIA). The work of the second author was funded by the Slovenian Research Agency through a young researcher grant. We wish to thank Jan Kralj for his insightful comments on the formulation of the proposed framework and for mathematical proofreading. Further, we are grateful to Vid Podpečan and Nika Eržen for their help with the implementation of the new version of the PyRDM library. Finally, we would like to thank the anonymous reviewers for careful reading, many insightful observations, and the encouragements to expand the initial work.

Appendix A: Wordification example

The Wordification approach is illustrated on a modified and substantially simplified version of the well-known East-West Trains domain (Michie et al. 1994). Our input database consists of just two tables shown in Fig. 6, where we have only one east-bound and one west-bound train, each with just two cars with certain properties[11].

The TRAIN table is the main table and the trains are the instances, with a class label denoting the direction of the train (east of west). As Fig. 7 shows, a multiset (a bag) of features is generated for each train t1 and t5 with the class label appended to the resulting feature vector (bag of features). Both single features and conjunctive features are shown in this example.

[10] https://github.com/SkBlaz/PropStar.

[11] Note that in the experiments we use the standard version of the East-West Trains domain.

TRAIN

trainID	eastbound
t1	east
...	...
t5	west
...	...

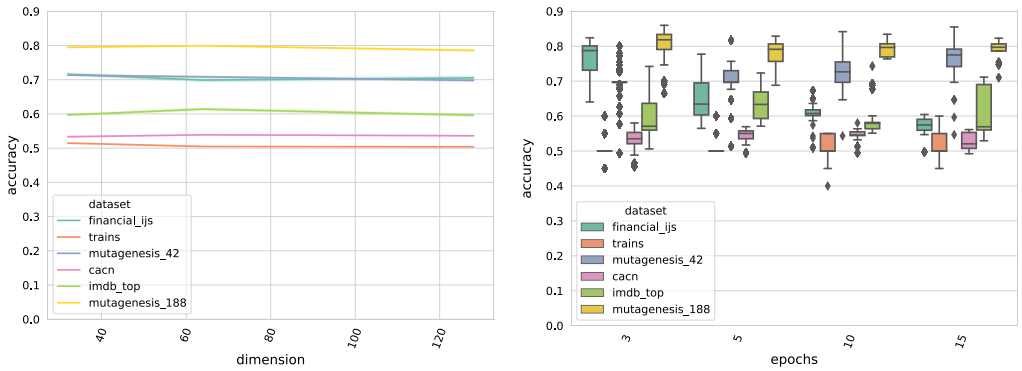
CAR

carID	shape	roof	train
c11	rectangle	none	t1
c12	rectangle	peaked	t1
...	...	...	...
c51	rectangle	none	t5
c52	hexagon	flat	t5
...	...	...	...

Fig. 6 Example input for Wordification in the East-West Trains domain

```
t1: {car_shape_rectangle, car_roof_none, car_shape_rectangle__car_roof_none,
     car_shape_rectangle, car_roof_peaked, car_shape_rectangle__car_roof_peaked}, east
    ...
t5: {car_shape_rectangle, car_roof_none, car_shape_rectangle__car_roof_none,
     car_shape_hexagon, car_roof_flat, car_shape_hexagon__car_roof_flat}, west
    ...
```

Fig. 7 The database from Fig. 6 in the bag-of-features representation (as in the original Wordification implementation, conjunctions of features are denoted by a long underscore instead of ∧)



(a) Dependence on hidden layer size. **(b)** Dependence on the number of epochs.

(c) Dependence on the learning rate. **(d)** Dependence on Dropout.

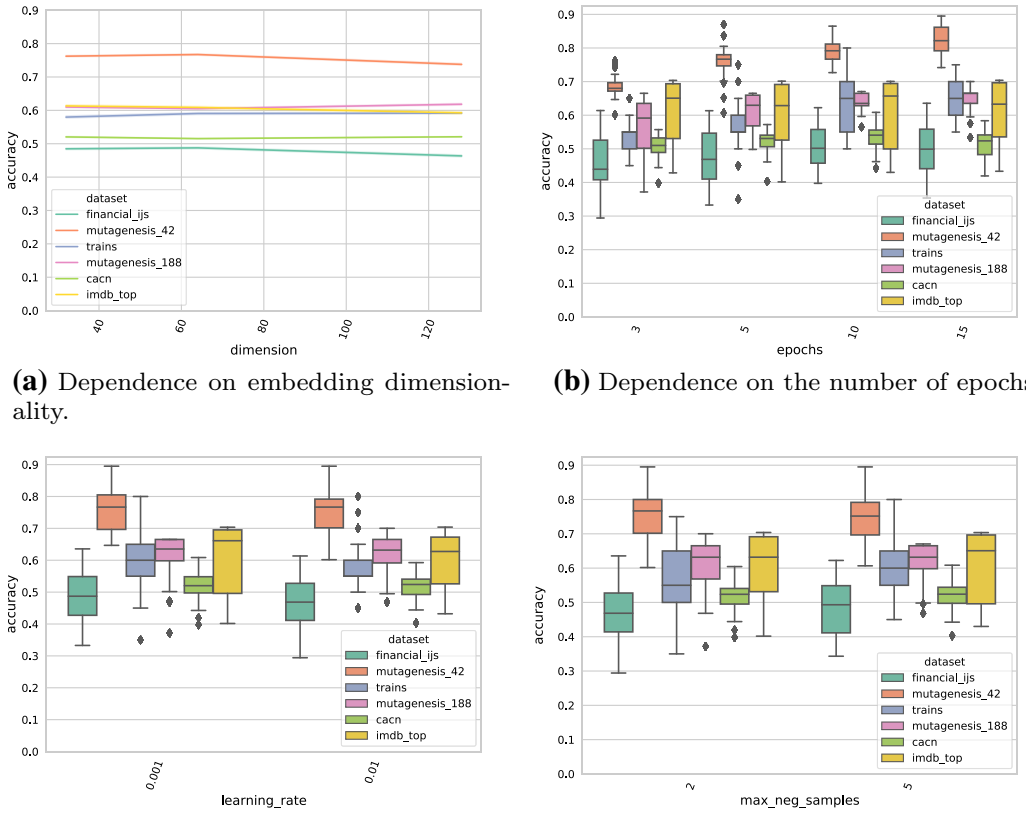
Fig. 8 Sensitivity of PropDRM to hyperparameter settings

Machine Learning (2020) 109:1465–1507 1501

Appendix B: Ablation study—PropDRM

We discuss the impact of individual hyperparameters on the performance of PropDRM. We first visualize the performance of PropDRM w.r.t. individual hyperparameters in Fig. 8.

We can observe that the relevance of individual hyperparameters varies from data set to data set. The learning rate, when too small, decreases the performance. In terms of embedding dimension, even smaller dimensions are sufficient for the considered data sets. This result potentially implies that the considered data sets are relatively small and contain only a small set of relevant features (when propositionalized). Thus, if the neural network detects the correct features as relevant, not many parameters are needed for a successful classification. An alternative explanation would imply that PropDRM learns hierarchical representations efficiently, albeit not optimized with their hierarchical nature in mind, which was previously demonstrated to capture hierarchical relations well (Nickel and Kiela 2017).



(a) Dependence on embedding dimensionality.

(b) Dependence on the number of epochs.

(c) Dependence on the learning rate.

(d) Dependence on the maximum negative sampling number.

Fig. 9 Sensitivity of PropStar to various hyperparameter settings

Appendix C: Ablation study—PropStar

We first explore the behavior with respect to various hyperparameter settings and visualize them in Fig. 9. We can observe that the amount of negative samples (Subfigure 9d)) impacts the PropStar's performance the most on the mutagenesis 42 data set, overall reducing the performance, even though a handful of models (outliers marked as dots) perform well. This indicates the importance of negative sample selection. As StarSpace does not use any sophisticated technique for sampling negative examples, the variability in performance could be notable due to this parameter.

It can be observed that a relatively small relational item embedding dimensionality is needed for successful performance. The dependence on other parameters varies from data set to data set. For example, the learning rate does not impact the larger Mutagenesis data set (Mut188) as much as it does the Trains data set. As the proposed methodology is not well adapted to such small data sets (e.g., tens of instances), large variability in performance could be linked to potential overfitting. Further, sufficient number of epochs are needed for PropStar to converge on individual data sets.

Appendix D: Interpretability of embedding-based methods using SHAP

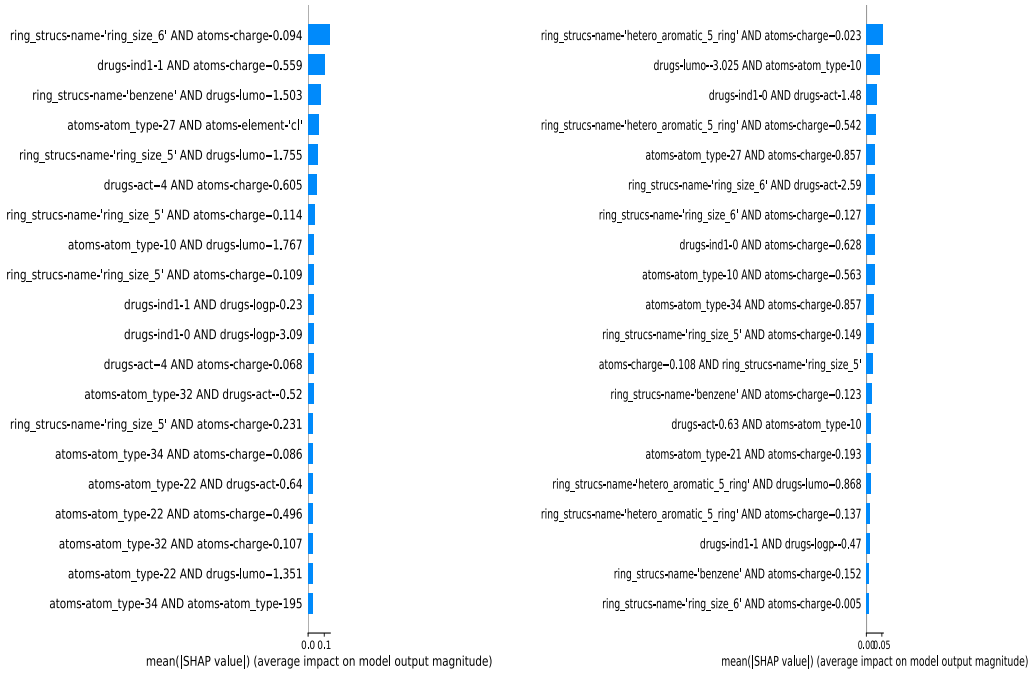
The approximation power of deep neural network which are commonly used with embeddings comes at a cost of lesser interpretability. Compared to symbolic relational (or propositional) learners, one cannot understand the deep relational models' deductive process by inspecting the model. However, *post hoc* explanation methods for prediction models can be used to better understand which parts of the feature space are relevant for the neural network's individual predictions. In this work, we leverage the state-of-the-art explanation tool SHAP (Lundberg and Lee 2017), based on the coalitional game theory. This tool captures the importance of interactions between features with Shapley values.

When considered in a feature importance scenario, the contribution of the i-th instance τ_i, is approximated by SHAP with the following expression:

$$\tau_i = \underbrace{\sum_{S \subseteq F \setminus \{i\}} \frac{|S|!(|F| - |S| - 1)!}{|F|!}}_{\text{All possible subsets}} \underbrace{\left[f(x_{S \cup \{i\}}) - f(x_S)\right]}_{\text{Difference in predictive performance}}$$

where S is a subset of all features F, f is the used predictive model, and x_S is an instance containing only features from the subset S. Shapley valufs offer insights into instance-level predictions by assigning fair credit to individual features for participation in interactions. They are commonly used to understand and debug black-box models.

In this work, we use the SHAP kernel approximator, the recently introduced, model-agnostic method for explaining model outputs. The used SHAP kernel explainer is considered an additive feature attribution method. Such methods are characterized as having an explanation model g that is a linear function of binary variables:



(a) Mean of SHAP explanations over the instances for PropDRM.

(b) Mean of SHAP explanations over the instances for PropStar.

Fig. 10 SHAP Kernel explanations of the two developed approaches

$$g(z') = \phi_0 + \sum_{i=1}^{|F|} \phi_i \cdot z'_i$$

where $z' \in \{0,1\}^{|F|}$, $|F|$ is the number of input features and $\phi_i \in \mathbb{R}$. This class of models assign an effect ϕ_i to each feature, and summing the effects of all such feature attributions approximates the output $f(x)$ of the original model. Detailed theoretical analysis of how this idea can be extended to approximation of outputs via a kernel is given in Lundberg and Lee (2017).

As an example demonstrating the explainability of the two paradigms, we visualize the Shapley values as explanations of learned classifiers for Mutagenesis 188 problem in Fig. 10. Explanations indicate parts of the feature space that have the largest impact on the model's output. Even though the considered SHAP kernel explainer is known to be a computationally expensive variant of SHAP (it is also the most general one), explanations were obtained in the order of minutes, indicating the potential of this methodology for explanations of predictors in larger relational databases.

References

Ahmed, C. F., Lachiche, N., Charnay, C., Jelali, S. E., & Braud, A. (2015). Flexible propositionalization of continuous attributes in relational data mining. *Expert Systems with Applications*, *42*(21), 7698–7709.

Benavoli, A., Corani, G., Demšar, J., & Zaffalon, M. (2017). Time for a change: A tutorial for comparing multiple classifiers through Bayesian analysis. *Journal of Machine Learning Research*, *18*(1), 2653–2688.

Bennett, K. P., Buja, A., Freund, W. S. Y., Schapire, R. E., Friedman, J., Hastie, T., et al. (2008). Responses to [52]. *Journal of Machine Learning Research*, *9*, 157–194.

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet allocation. *Journal of Machine Learning Research*, *3*(Jan), 993–1022.

Blockeel, H., Raedt, L. D., & Ramon, J. (1998). Top-down induction of clustering trees. In *Proceedings of the 15th international conference on machine learning*, pp. 55–63. Morgan Kaufmann.

Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., & Yakhnenko, O. (2013). Translating embeddings for modeling multi-relational data. *Advances in Neural Information Processing Systems*, pp. 2787–2795.

Bordes, A., Glorot, X., Weston, J., & Bengio, Y. (2014). A semantic matching energy function for learning with multi-relational data. *Machine Learning*, *94*(2), 233–259.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*(2), 123–140.

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32.

Breiman, L., Friedman, J. H., Olshen, R., & Stone, C. (1984). *Classification and regression trees*. Pacific Grove, CA: Wadsworth & Brooks.

Chang, S., Han, W., Tang, J., Qi, G. J., Aggarwal, C. C., & Huang, T. S. (2015). Heterogeneous network embedding via deep architectures. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 119–128. ACM.

Charnay, C., Lachiche, N., & Braud, A. (2015). CARAF: Complex aggregates within random forests. In *Inductive logic programming—25th international conference, ILP 2015*, Kyoto, Japan, August 20–22, 2015, Revised Selected Papers, pp. 15–29. Springer.

Clark, P., & Niblett, T. (1989). The CN2 induction algorithm. *Machine Learning*, *3*(4), 261–283.

Clevert, D. A., Unterthiner, T., & Hochreiter, S. (2016). Fast and accurate deep network learning by exponential linear units (ELUs). In *International conference on representation learning, ICLR*. arXiv :1511.07289.

Corani, G., Benavoli, A., Demšar, J., Mangili, F., & Zaffalon, M. (2017). Statistical comparison of classifiers through Bayesian hierarchical modelling. *Machine Learning*, *106*(11), 1817–1837.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273–297.

Cumby, C. M., & Roth, D. (2003). On kernel methods for relational learning. In *Proceedings of the 20th international conference on machine learning (ICML-03)*, pp. 107–114.

Dash, T., Srinivasan, A., Vig, L., Orhobor, O. I., & King, R. D. (2018). Large-scale assessment of deep relational machines. In *Proceedings of the international conference on inductive logic programming*, pp. 22–37. Springer, Berlin.

Dash, T., Srinivasan, A., Joshi, R. S., & Baskar, A. (2019). Discrete stochastic search and its application to feature-selection for deep relational machines. In I. V. Tetko, V. Kůrková, P. Karpov, & F. Theis (Eds.), *Artificial neural networks and machine learning: ICANN 2019–deep Learning* (pp. 29–45). Berlin: Springer.

De Raedt, L. (2008). *Logical and relational learning*. Berlin: Springer.

Debnath, A. K., Lopez de Compadre, R. L., Debnath, G., Shusterman, A. J., & Hansch, C. (1991). Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. Correlation with molecular orbital energies and hydrophobicity. *Journal of Medicinal Chemistry*, *34*(2), 786–797.

Demeester, T., Rocktäschel, T., & Riedel, S. (2016). Lifted rule injection for relation embeddings. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pp. 1389–1399.

Dumančić, S., Guns, T., Meert, W., & Blockleel, H. (2018). Auto-encoding logic programs. In *Proceedings of the international conference on machine learning*, Stockholm, Sweden.

Džeroski, S., & Lavrač, N. (Eds.). (2001). *Relational data mining*. Berlin: Springer.

Flach, P., & Lachiche, N. (1999). 1BC: A first-order Bayesian classifier. In *International conference on inductive logic programming*, pp. 92–103. Berlin: Springer.

Flach, P., & Lachiche, N. (2001). Confirmation-guided discovery of first-order rules with Tertius. *Machine Learning*, *42*(1/2), 61–95.

Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, *55*(1), 119–139.

Friedman, J. H., & Fisher, N. I. (1999). Bump hunting in high-dimensional data. *Statistics and Computing*, *9*(2), 123–143.

Gärdenfors, P. (2000). *Conceptual spaces: The geometry of thought*. Cambridge, MA: MIT Press.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. Cambridge: MIT Press.

Grčar, M., Trdin, N., & Lavrač, N. (2013). A methodology for mining document-enriched heterogeneous information networks. *The Computer Journal*, *56*(3), 321–335.

Grover, A., & Leskovec, J. (2016). node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 855–864.

Guo, S., Wang, Q., Wang, L., Wang, B., & Guo, L. (2016). Jointly embedding knowledge graphs and logical rules. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pp. 192–202.

Haussler, D. (1999). Convolution kernels on discrete structures. Tech. rep., Department of Computer Science, University of California.

He, S., Liu, K., Ji, G., & Zhao, J. (2015). Learning to represent knowledge graphs with Gaussian embedding. In *Proceedings of the 24th ACM international on conference on information and knowledge management*, pp. 623–632. ACM.

Kralj, J., Robnik-Šikonja, M., & Lavrač, N. (2018). HINMINE: Heterogeneous information network mining with information retrieval heuristics. *Journal of Intelligent Information Systems*, *50*(1), 29–61.

Kramer, S., Lavrač, N., & Flach, P. (2001). Propositionalization approaches to relational data mining. In S. Džeroski & N. Lavrač (Eds.), *Relational data mining* (pp. 262–291). Berlin: Springer.

Krogel, M. A., & Wrobel, S. (2001). Transformation-based learning using multirelational aggregation. In *Proceedings of international conference on inductive logic programming*, pp. 142–155. Berlin: Springer.

Krogel, M. A., Rawles, S., Železný, F., Flach, P., Lavrač, N., & Wrobel, S. (2003). Comparative evaluation of approaches to propositionalization. In T. Horvath & A. Yamamoto (Eds.), *Proceedings of the 13th international conference on inductive logic programming (ILP-2003* (pp. 197–214). Berlin: Springer.

Kuželka, O., & Železný, F. (2008). HiFi: Tractable propositionalization through hierarchical feature construction. In Železný, F., Lavrač, N. (Eds.) *Late breaking papers, the 18th international conference on inductive logic programming*, pp. 69–74.

Kuželka, O., & Železný, F. (2011). Block-wise construction of tree-like relational features with monotone reducibility and redundancy. *Machine Learning*, *83*(2), 163–192.

Lachiche, N., & Flach, P. A. (2003). 1BC2: A true first-order Bayesian classifier. *Proceedings of inductive logic programming*, pp. 133–148.

Lavrač, N., Džeroski, S., & Grobelnik, M. (1991). Learning nonrecursive definitions of relations with LINUS. In *Proceedings of the 5th European working session on learning (EWSL-91)*, pp. 265–281. Springer, Porto, Portugal.

Lavrač, N., Kralj Novak, P., Mozetič, I., Podpečan, V., Motaln, H., Petek, M., & Gruden, K. (2009). Semantic subgroup discovery: Using ontologies in microarray data analysis. In *Proceedings of the 31st annual international conference of the IEEE EMBS*, pp. 5613–5616.

Lavrač, N., & Džeroski, S. (1994). *Inductive logic programming: Techniques and applications*. New York: Ellis Horwood.

Lavrač, N., & Flach, P. (2001). An extended transformation approach to inductive logic programming. *ACM Transactions on Computational Logic*, *2*(4), 458–494.

Le, Q., & Mikolov, T. (2014). Distributed representations of sentences and documents. In *Proceedings of international conference on machine learning*, pp. 1188–1196.

Lewis, D. D. (1992). An evaluation of phrasal and clustered representations on a text categorization task. In *Proceedings of the 15th annual international ACM SIGIR conference on research and development in information retrieval*, pp. 37–50 .

Lodhi, H. (2013). Deep relational machines. In *Proceedings of the international conference on neural information processing*, pp. 212–219. Berlin: Springer.

Lundberg, S. M., & Lee, S. I. (2017). A unified approach to interpreting model predictions. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (Eds.) *Advances in neural information processing systems*, pp. 4765–4774.

McInnes, L., Healy, J., Saul, N., & Grossberger, L. (2018). UMAP: Uniform manifold approximation and projection. *The Journal of Open Source Software*, *3*(29), 861.

Mease, D., & Wyner, A. (2008). Evidence contrary to the statistical view of boosting. *Journal of Machine Learning Research*, *9*, 131–156.

Michalski, R. S., Mozetič, I., Hong, J., & Lavrač, N. (1986). The multi-purpose incremental learning system AQ15 and its testing application on three medical domains. In *Proceedings of the 5th national conference on artificial intelligence*, pp. 1041–1045. Philadelphia, PA.

Michie, D., Muggleton, S., Page, D., & Srinivasan, A. (1994). To the international computing community: A new East-West challenge. Tech. rep., Oxford University Computing laboratory, Oxford, UK.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z.

Ghahramani, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems 26* (pp. 3111–3119). New York, USA: Curran Associates Inc.

Motl, J., & Schulte, O. (2015). The CTU Prague relational learning repository. arXiv:1511.03086.

Muggleton, S. H. (Ed.). (1992). *Inductive logic programming*. London: Academic Press Ltd.

Muggleton, S. (1995). Inverse entailment and Progol. *New Generation Computing*, *13*(3–4), 245–286.

Nickel, M., & Kiela, D. (2017). Poincaré embeddings for learning hierarchical representations. In *Advances in neural information processing systems*, pp. 6338–6347.

Nickel, M., Tresp, V., & Kriegel, H. P. (2011). A three-way model for collective learning on multi-relational data. *Proceedings of International Conference on Machine Learning*, *11*, 809–816.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al. (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, *12*(Oct), 2825–2830.

Perovšek, M., Vavpetič, A., Cestnik, B., & Lavrač, N. (2013). A wordification approach to relational data mining. In *Proceedings of the international conference on discovery science*, pp. 141–154. Berlin: Springer.

Perovšek, M., Vavpetič, A., Kranjc, J., Cestnik, B., & Lavrač, N. (2015). Wordification: Propositionalization by unfolding relational data into bags of words. *Expert Systems with Applications*, *42*(17–18), 6442–6456.

Perozzi, B., Al-Rfou, R., & Skiena, S. (2014). Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 701–710. ACM.

Plantié, M., & Crampes, M. (2013). Survey on social community detection. In N. Ramzan, R. Zwol, J. S. Lee, K. Clüver, & X. S. Hua (Eds.), *Social media retrieval* (pp. 65–85). London: Springer.

Podpečan, V., Lavrač, N., Mozetič, I., Kralj Novak, P., Trajkovski, I., Langohr, L., et al. (2011). SegMine workflows for semantic microarray data analysis in Orange4WS. *BMC Bioinformatics*, *12*, 416.

Qiu, J., Dong, Y., Ma, H., Li, J., Wang, K., & Tang, J. (2018). Network embedding as matrix factorization: Unifying DeepWalk, LINE, PTE, and Node2Vec. In *Proceedings of the eleventh ACM international conference on web search and data mining*, WSDM '18, pp. 459–467. ACM.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, *1*(1), 81–106.

Ribeiro, L. F., Saverese, P. H., & Figueiredo, D. R. (2017). Struc2vec: Learning node representations from structural identity. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining, KDD '17*, pp. 385–394. New York: ACM.

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). Why should I trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1135–1144. ACM.

Ristoski, P., & Paulheim, H. (2016). Rdf2vec: Rdf graph embeddings for data mining. In P. Groth, E. Simperl, A. Gray, M. Sabou, M. Krötzsch, F. Lecue, F. Flöck, & Y. Gil (Eds.), *The semantic web: ISWC 2016* (pp. 498–514). Cham: Springer.

Robnik-Šikonja, M., & Kononenko, I. (2008). Explaining classifications for individual instances. *IEEE Transactions on Knowledge and Data Engineering*, *20*(5), 589–600.

Rocktäschel, T., Singh, S., & Riedel, S. (2015). Injecting logical background knowledge into embeddings for relation extraction. In *Proceedings of the 2015 conference of the north American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1119–1129.

Rumelhart, D. E., & McClelland, J. L. (Eds.) (1986). Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1: Foundations. MIT Press, Cambridge, MA.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533.

Schapire, R. E., Freund, Y., Bartlett, P., & Lee, W. S. (1998). Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, *26*(5), 1651–1686.

Schölkopf, B., & Smola, A. J. (2001). *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. Cambridge: The MIT Press.

Škrlj, B., Kralj, J., Konc, J., Robnik-Šikonja, M., & Lavrač, N. (2019). Deep node ranking: An algorithm for structural network embedding and end-to-end classification. arXiv:1902.03964.

Srinivasan, A. (2007). Aleph manual. http://www.cs.ox.ac.uk/activities/machinelearning/Aleph/.

Srinivasan, A., King, R. D., Muggleton, S., & Sternberg, M. J. (1997). Carcinogenesis predictions using ILP. In *Proceedings of the international conference on inductive logic programming*, pp. 273–287. Berlin: Springer.

Srinivasan, A., Vig, L., & Bain, M. (2019). Logical explanations for deep relational machines using relevance information. *Journal of Machine Learning Research*, *20*(130), 1–47.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, *15*(1), 1929–1958.

Štrumbelj, E., & Kononenko, I. (2014). Explaining prediction models and individual predictions with feature contributions. *Knowledge and Information Systems*, *41*(3), 647–665.

Tang, J., Qu, M., & Mei, Q. (2015a). PTE: Predictive text embedding through large-scale heterogeneous text networks. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1165–1174. ACM.

Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., & Mei, Q. (2015b). LINE: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*, pp. 1067–1077.

Van Der Walt, S., Colbert, S. C., & Varoquaux, G. (2011). The NumPy array: A structure for efficient numerical computation. *Computing in Science & Engineering*, *13*(2), 22.

Vapnik, V. (1995). *The nature of statististical learning theory*. New York: Springer.

Vavpetič, A., & Lavrač, N. (2011). Semantic data mining system g-SEGS. In *Proceedings of the workshop on planning to learn and service-oriented knowledge discovery (PlanSoKD-11), ECML PKDD conference*, pp. 17–29.

Wang, Q., Wang, B., & Guo, L. (2015). Knowledge base completion using embeddings and rules. In *Proceedings of the 24th international joint conference on artificial intelligence*, pp. 1859–1865.

Wang, Z., Zhang, J., Feng, J., & Chen, Z. (2014). Knowledge graph and text jointly embedding. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1591–1601.

Wang, Q., Mao, Z., Wang, B., & Guo, L. (2017). Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, *29*(12), 2724–2743.

Wu, L. Y., Fisch, A., Chopra, S., Adams, K., Bordes, A., & Weston, J. (2018). Starspace: Embed all the things! In *Proceedings of the 32nd AAAI conference on artificial intelligence*, pp. 5569–5577.

Železný, F., & Lavrač, N. (2006). Propositionalization-based relational subgroup discovery with RSD. *Machine Learning*, *62*, 33–63.

Zhu, S., Bing, J., Min, X., Lin, C., & Zeng, X. (2018). Prediction of drug–gene interaction by using metapath2vec. *Frontiers in Genetics, 9*.

Žitnik, M., & Leskovec, J. (2017). Predicting multicellular function through multi-layer tissue networks. *Bioinformatics*, *33*(14), i190–i198.

Affiliations

Nada Lavrač[1,2] · Blaž Škrlj[3] · Marko Robnik-Šikonja[4]

Nada Lavrač
nada.lavrac@ijs.si

Marko Robnik-Šikonja
marko.robnik@fri.uni-lj.si

[1] Jožef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia

[2] University of Nova Gorica, Glavni trg 8, 5271 Vipava, Slovenia

[3] International Postgraduate School Jožef Stefan, Jožef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia

[4] University of Ljubljana, Faculty of Computer and Information Science, Večna pot 113, 1000 Ljubljana, Slovenia