Regularising LSTM classifier by transfer learning for detecting misogynistic tweets with small training set

Md Abul Bashar1 · Richi Nayak1 · Nicolas Suzor2

Received: 4 August 2019 / Accepted: 6 June 2020
© Springer-Verlag London Ltd., part of Springer Nature 2020

Abstract
Supervised machine learning methods depend highly on the quality of the training dataset and the underlying model. In particular, neural network models, that have shown great success in dealing with natural language problems, require a large dataset to learn a vast number of parameters. However, it is not always easy to build a large (labelled) dataset. For example, due to the complex nature of tweets and the manual labour involved, it is hard to create a large Twitter data set with the misogynistic label. In this paper, we propose to regularise a long short-term memory (LSTM) classifier using a pretrained LSTM-based language model (LM) to build an accurate classification model with a small training set. We explain transfer learning (TL) with a Bayesian interpretation and show that TL can be viewed as an uncertainty regularisation technique in Bayesian inference. We show that a LM pre-trained on a sequence of general to task-specific domain datasets can be used to regularise a LSTM classifier effectively when a small training dataset is available. Empirical analysis with two small Twitter datasets reveals that an LSTM model trained in this way can outperform the state-of-the-art classification models.

Keywords Misogynistic tweet · Transfer learning · LSTM · Small dataset · Overfitting

1 Introduction

Incidents of abuse, hate and harassment have grown with the proliferated use of social media platforms (e.g. Twitter, Facebook, Instagram) [14]. Online abuse targeting women (i.e. name-calling, offensive language, threats of harm or sexual violence, intimidation, shaming, and silencing) has become common [2]. An automated method to identify

---

1 School of Computer Science, Queensland University of Technology, Brisbane, Australia
2 School of Law, Queensland University of Technology, Brisbane, Australia
misogynistic tweets can help in ongoing efforts to develop effective remedies [53]. A key challenge in misogynistic tweet detection is understanding the context of a tweet. The complex and noisy nature of tweets makes it difficult.

Classification of a tweet based on the presence of typical misogynistic keywords using a lexical detection approach results in poor accuracy [11, 60]. Simple methods such as bag-of-words, part-of-speech (POS) and belief propagation perform poorly due to the noise present in the data [60]. An algorithm should identify patterns, sequences, and other complex features that can be correlated with misogynistic tweets despite the noise.

In the supervised machine learning setting, traditional algorithms such as Random Forest [32] and Support Vector Machines [21] rely on manual processing to obtain good features and are limited by the kinds of features available. On the other hand, neural network (NN) models can learn complex features and effectively use them to classify a given instance [61]. In the last few years, long short-term memory (LSTM) [22], a special kind of recurrent neural network (RNN) [46], has shown to be effective in natural language processing (NLP) tasks such as text classification [24], language modelling [12, 51], machine translation [52], and question answering [57]. The success of LSTM can be credited to its capability to capture the context from the long-term feature dependencies of a sequence [22].

Given the requirement of setting a huge number of parameters, LSTM-based classification models need to be trained on a huge labelled dataset. If the training dataset is small (e.g. misogynistic tweets), the model parameters overfit the dataset. In general, this is a limitation of any NN-based model [15]. To address the overfitting problem, we investigate the effectiveness of transfer learning (TL) [12, 24] by building a language model (LM) on a number of datasets ranging from general to specific domains that can capture the features dependencies without overfitting.

To understand the statistical view of transfer learning, we present a Bayesian interpretation [6, 17, 36] of TL and show that LM can be viewed as a regularisation technique. Bayesian interpretation is used to analyse and estimate uncertainties in TL, e.g. uncertainty in the pretraining datasets and pretraining models. This helps to choose right datasets and models for pretraining. The finding is not limited to LSTM; it can apply to other models such as convolutional neural network (CNN) and feed forward neural network.

We then propose a novel LSTM-based Bayesian transfer learning method and extend it to be used with the LSTM classifier, LSTM-based LM regularised classifier (LSTM-L), for detecting misogynistic tweets with a small training set. The theoretical analyses and experimental results show that LSTM-based LM is significantly better than TL through Word2Vec [42] for controlling the overfitting problem. Results show that the classification performance of LSTM can be improved by using the pretrained LM with multiple datasets from general to specific domain, when only a small size training data are available for the downstream task.

In general, this paper investigates how to effectively apply deep learning methods, with a focus on reducing overfitting, on an unbalanced text data set such as misogynistic tweets. Though we present the proposed method for the problem of misogynistic tweet detection, the method is applicable to any other situation with similar issues.

The specific contributions of this paper are as follows. (1) It shows that a LSTM-based LM pretrained on datasets from general to task-specific domain can be used to effectively train a LSTM classifier when a small set of labelled data is available. (2) It gives a Bayesian interpretation of TL and shows that TL can be viewed as a technique of regularising overfitting (or uncertainty) in a classification model. (3) It presents an automated mining method to detect misogynistic abusive message on social media platforms.
To our best of knowledge, this will be the first work that extensively investigates TL of LM using multiple datasets and proposes a novel method of misogynistic tweet detection based on such TL in LM. The rest of the paper is organised as follows: Sect. 2 discusses related work and background; problem formulation is given in Sect. 3; Sect. 4 discusses misogynistic tweet classification regularised by transfer learning; Sect. 5 provides the detailed evaluation and discussion of the model, and the paper concludes in Sect. 6.

2 Related work

Misogynistic abusive tweet detection falls into the general research area of text classification. Lack of unique and discriminative linguistic characteristics in misogynistic and hate speech makes it difficult to separate them from non-misogynistic and non-hate speech [58, 63], as shown in Table 1. Social media users commonly use offensive words or expletives in their online dialogue to express satire and jokes [58]. Improving the training dataset such as increasing the size, reducing class imbalance, performing over-sampling and under-sampling may not always guarantee a separation between two classes for machine learners. Machine learning models cannot just rely on the occurrences of abusive words; they need to incorporate the context of these words in learning. For example, individual words such as muslim, refugee, and troublemakers are not always indicative features of hate speech but their combinations can be more indicative [63].

Feature representation of the bag-of-words (BOW) approach is straightforward that can achieve high recall. However, it results in higher number of false positives by misclassifying tweets as abusive due to the presence of misogynistic words [27]. Researchers attempted to capture the context by using biagrams with unigrams [1, 43]; however, they only achieved marginal improvement. Using higher grams may have adverse affect due to their low frequency [30]. These methods are further challenged by the lack of examples of misogynistic tweets in a dataset due to the difficulty of obtaining a balanced training dataset.

Previous work has leveraged syntactic features and the intensity of hate speech to identify good features for use in classification [1, 19, 48]. A number of methods were proposed in the task of detecting misogynistic tweets in AMI@IberEval shared task-A [1]. Participants used individual algorithm as well ensembles based on logistic regression, support vector machine (SVM), random forest, gradient boosting and stochastic gradient descent. Researchers explored traditional techniques (i.e. bag-of-words/characters and unigrams and

| Table 1 Examples of Misogynistic (label 1) and Non-misogynistic (label 0) Tweets. xatp is used to anonymise @ someone in the tweet |
|---|---|
| Label | Tweet |
| 1 | xatp i hope you choke you whore |
| 1 | name a bigger slut than xatp |
| 1 | xatp shut up whore |
| 0 | xatp sorry Master I m a stupid and pathetic whore |
| 0 | xatp xatp Nonsense, that’s like saying someone is intolerant of murder or rape. |
| 0 | xatp I love you slut |
| 0 | xatp Life’s a slut |
| 0 | xatp Mmm. Whore burger |
bigrams) as well as specific lexical features for representing the tweet contents. Methods that spent efforts on manual feature engineering were able to improve the performance. The highest performing team used vector representation that concatenates sentence embedding, weighted BOW and average word embeddings coupled with a logistic regression model.

Performance of traditional algorithms highly depends on feature engineering and feature representation [11, 60]. On the other hand, NN-based classifiers have become popular due to their ability to automatically learn abstract features from the given input feature representation and reduced dependency on manual feature engineering [3]. For example, a deep neural network (DNN) model has shown to extract discriminative features that can capture semantics of hate speech [63]. Input to NN-based algorithms can be various forms of feature encoding, including those used in the classic methods. Algorithm design in this category focuses on the selection of the network topology to automatically extract useful abstract features. Popular network architectures are CNN, RNN and LSTM. CNN is well known for extracting patterns similar to phrases and nGrams [3]. On the other hand, RNN and LSTM have been found effective for sequence learning such as order information in text [3] and have been successful in text classification [26].

In AMI@IberEval shared task-A [1], a DNN was adopted by a team that achieved a moderate performance. An ensemble of RNN classifiers has been used to identify hateful content in social media [44]. The inputs to the classifiers included not only features in the tweets but also the features of the users such as users’ tendency towards racism or sexism. NN classifier models typically require a huge amount of labelled data for training; otherwise, they can overfit the training data [50]. Random dropout [49], L1 and L2 regularisation are commonly used to regularise NN architecture. These regularisations reduce overfitting for medium size datasets.

NN models overfit to small datasets [18]. Researchers have used data augmentation and transfer learning to address this problem. Data augmentation is achieved by expanding documents by adding or replacing semantically similar words using word embedding [54]. However, this is not effective in tweet classification due to the noisy nature of the dataset. Our experimental results in Sect. 5.4 also confirm this. In text processing, the simplest form of transfer learning has been used by generating features from the pretrained word embeddings (also called pretrained word vectors) [55], sentence embeddings [34] and paragraph embeddings to be used in the downstream model of a NLP task. These pretrained embeddings provide improvement over embeddings learned from only the target data [5].

An emerging trend is pretraining a language model (LM) on a similar large-scale dataset [10, 12, 24, 45], and then, fine-tune the model to a supervised downstream task such as classification. The top performing team named NULI [33] in SemEval-2019 [33] used the pretrained transformer (BERT) for detecting offensive language on social media.

In this paper, we propose to pretrain a LSTM-based LM with multiple datasets and fine-tune the model to the misogynistic tweet detection task. The use of multiple datasets in TL has not been investigated before. It remains to be seen what are their effects and dependencies.

Firstly, we will investigate whether TL can be sensitive to the domain knowledge of the pretraining corpus. Should a LM needed to be pretrained from a general to specific domain to capture contextual properties? Secondly, there exist several NN models and possible TL options. Because of different underlying assumptions and varying architectures, the right choice heavily depends on the problem at hand. For example, TL using CNN has been found effective in image classification [47], whereas TL using LSTM-based LM has been found effective in text classification [24]. As Word2Vec [42] is a linear model and linear algebra can be applied to the vectors pretrained using this model, it is effective in automatic
analogy computation. However, LSTM-based LM cannot be used in analogy computation because of nonlinearity. Therefore, an effective TL technique is needed to develop to classify misogynistic tweets accurately.

Thirdly, it requires a rigorous investigation as to what extent TL through LSTM-based LM can reduce overfitting of a classification model. Lastly, it is useful to have a statistical understanding of TL using LM in deep learning as transfer learning is considered a black box approach. A statistical understanding will help to comprehend why TL using LM in deep NNs is working, identify the potential application areas and investigate for future improvements. There are some studies on statistical understanding on neural networks [6, 17, 31, 36]; however, there exists no study on transfer learning using LM in deep NN.

3 Problem formulation: misogynistic tweet classification

In this paper, we focus on distinguishing tweets that contain misogynistic words as abusive and non-abusive and identifying misogynistic tweets that are abusive towards an individual or a group. They are a subset of the larger category of tweets that include sexist words or concepts. Not all tweets that contain misogynistic keywords are abusive; these words have been commonly used in non-abusive tweets too (e.g. Table 1). Accordingly, we address the difficult task of separating abusive tweets from the tweets that are sarcastic, joking, or contained misogynistic keywords in non-abusive contexts.

We used a systematic approach to generate the labelled data manually. The following contextual information is checked to label a tweet as targeted misogynistic abuse: (a) Is a specific person or group being targeted in this tweet? (b) Does this tweet contain a specific threat or wish for violence? (c) Does this tweet encourage or promote self-harm or suicide? (d) Is the tweet harassing a specific person, or inciting others to harass a specific person? (e) Does the tweet use misogynistic language in objectifying a person, making sexual advances, or sending sexually explicit material? (f) Is the tweet promoting hateful conduct by gender, sexual orientation, etc.?

The labelled tweets reveal many challenges that need to be addressed to train a classifier effectively. These include: (a) The misogynistic words are not the discriminatory words. They appear both in misogynistic and non-misogynistic tweets. (b) The data is noisy and does not follow a standard language sequence (format). Words may be misspelt or spelt in many ways. Some tweets contain words from local dialects or foreign languages. (c) Detecting misogynistic tweets needs the semantics and context information that is often not available, e.g. it is difficult to use the dictionary-based semantics due to the noise (abbreviations, misspelling, unknown words, etc.), and it is difficult to infer the context due to small length of a tweet. (d) The labelling process is time-consuming and is extremely difficult to generate a large quantity of labelled data.

Given these challenges, we investigate how to effectively train a LSTM classifier with a small set of labelled data to detect misogynistic abusive tweets. We propose a novel three-step method to approach this problem:

- **Prefiltering** Filter tweets that contain any of the three main misogynistic keywords (slut, rape, whore) to find potential misogynistic tweets [4]. Without this step, it requires reading through thousands of tweets to identify a potential misogynistic tweet.

- **Pretraining language model** Train a LSTM-based LM with a sequence of unlabelled large datasets from general to specific domains to learn the language patterns.
Training classifier with TL: Train a LSTM classifier, which is regularised by the pretrained LM, using a small labelled data set to detect the misogynistic tweets. We propose to apply TL by training a LSTM classifier on top of a pretrained LM.

We investigate the followings. (1) What kind of pretraining datasets are effective in capturing semantic and contextual information for tweet classification? (2) Can the patterns, sequences and other linguistic properties identify misogynistic tweets where misogynistic words are not discriminatory? (3) How to capture both noise-patterns of tweets and linguistic properties using pretraining datasets to regularise the uncertainties of tweet classification model? (4) Why a specific pretraining approach succeed or fail to regularise a classifier?

4 Proposed method of classifier regularised by transfer learning

Figure 1 shows the process of transfer learning for the proposed classification model through LM. A LM is pretrained with a sequence of unlabelled datasets from general to specific domain. The pretrained LSTM LM provides the prior knowledge, i.e. the initial weights. With using the labelled dataset, only the necessary weights are updated while freezing some of the weights. Since the labelled dataset does not contain a large number of instances, the trained model will rely on initial weights instead of overfitting the small labelled dataset. Once the tuning of language model is completed with the small dataset, the trained LSTM-L classifier is ready to predict previously unseen tweets.

In Sect. 4.1, we discuss the problem definition of regularising classifier with unlabelled data; in Sect. 4.2, we discuss unsupervised learning of Language Model with the goal of regularising classifiers. After that, we discuss neural network language model and transfer learning for language model in Sects. 4.3 and 4.4, respectively. Finally, we discuss regularisation of a classifier with LSTM-based LM in Sect. 4.5.

4.1 Problem definition: regularising classifier with unlabelled data

Let $X$ be a small size dataset that contains $n$ features and $K$ classes. Let $x = (x_1, \ldots, x_n)$ be a vector representing an instance in $X$. Let $C_k$ be a set of $K$ classes. The classification task is to assign an instance to a class $C_k$ based on the feature vector $x$, i.e. finding $p(C_k|x)$. 
A generative model learns the probability distribution \( p(x|C_k) \) and \( p(C_k) \) that can be transformed into \( p(C_k|x) \) by Bayes theorem [13]:

\[
p(C_k|x) = \frac{p(x|C_k)p(C_k)}{p(x)} = \frac{p(x, C_k)}{p(x)}
\]

Equation 1 can be rearranged as:

\[
p(x, C_k) = p(C_k|x)p(x)
\]

Probability \( p(x) \) in Eq. 2 can be seen as a regularizer for \( p(C_k|x) \) that can regularise modelling of the associated uncertainties of \( p(x, C_k) \). As \( p(x) \) does not depend on \( C_k \), this means that \( p(x) \) can be learned independent of class level \( C_k \). That is, \( p(x) \) can be learned from unlabelled data.

The objective of this paper is to investigate how much the estimation of \( p(x, C_k) \) can be improved when \( p(x) \) is learned from a sequence of unlabelled datasets. In other words, we study the possibility of improving the prediction accuracy of a (discriminative) classification model using the unlabelled data.

A discriminative model such as LSTM learns to classify an instance \( x \) into class \( C_k \) by learning the conditional probability distribution as \( p(C_k|x, \theta) \approx p(C_k|x) \) where \( \theta \) is the list of model parameters. However, accurately approximating \( p(C_k|x) \) requires a large number of labelled instances. If only a small set of labelled data is available, the learned \( p(C_k|x, \theta) \) might not be a good approximation of population distribution because \( \theta \) may over-fit the small training set. Alternately, \( p(x) \) can be learned from one or more large unlabelled datasets and conditioned on \( x \) to learn \( p(C_k|x, \theta) \), leading to \( p(C_k|x, \theta)p(x) \approx p(x, C_k) \).

The term \( p(C_k|x, \theta)p(x) \) can be seen as equivalent to combining the regularisation into the discriminative model. This regularised model would act similar to a generative model. A prior study has empirically showed that an intermediate model between generative and discriminative models often gives better accuracy [7]. We propose to learn \( p(x) \) from a sequence of unlabelled datasets and use it to learn \( p(C_k|x, \theta)p(x) \approx p(x, C_k) \) with a small training dataset. Next, we present the estimation of \( p(x) \) as a language model using unsupervised learning.

### 4.2 Constructing language model with unsupervised learning

Probability \( p(x) \) can be estimated using the assumption of Language model where features are considered conditionally dependent. This is to support natural language processing where in a sentence, the sequencing of words depends on each other. Based on this, the joint probability \( p(x, C_k) \) in Eq. 2 can be rewritten as follows, using the chain rule:
\[ p(\mathbf{x}, C_k) = p(C_k | \mathbf{x}) p(\mathbf{x}) \]
\[ = p(C_k | x_1, \ldots, x_n) p(x_1, \ldots, x_n) \]
\[ = p(C_k | x_1, \ldots, x_n) p(x_1 | x_2, \ldots, x_n) p(x_2, \ldots, x_n) \]
\[ = \ldots \]
\[ = p(C_k | x_1, \ldots, x_n) p(x_1 | x_2, \ldots, x_n) \ldots p(x_{n-1} | x_n) p(x_n) \]
\[ = p(C_k | x_1, \ldots, x_n) \prod_{i=1}^{n} p(x_i | x_1 \ldots x_{i-1}) \]
\[ = p(C_k | \mathbf{x}) \prod_{i=1}^{n} p(x_i | x_1 \ldots x_{i-1}) \]

The part \( \prod_{i=1}^{n} p(x_i | x_1 \ldots x_{i-1}) \) in Eq. 3 can be considered as a language model because \( p(x_i | x_1 \ldots x_{i-1}) \) seeks to predict the probability of observing the \( i \)th feature \( x_i \), given the previous \((i-1) \) features \((x_1 \ldots x_{i-1}) \). \( \prod_{i=1}^{n} p(x_i | x_1 \ldots x_{i-1}) = p(x_1, \ldots, x_n) = p(\mathbf{x}) \) can be interpreted as the probability of observing an instance (e.g. a sequence or a sentence) in a corpus. However, it is computationally difficult to estimate \( \prod_{i=1}^{n} p(x_i | x_1 \ldots x_{i-1}) \). A simple estimation for this can be:

\[ p(x_i | x_1, \ldots, x_{i-1}) = \frac{\text{count}(x_1, \ldots, x_{i-1}, x_i)}{\text{count}(x_1, \ldots, x_{i-1})} \]

Nonetheless, observing enough data (in order to obtain realistic counts for any sequence of \( i \) features for any nontrivial value of \( i \)) from a dataset is unrealistic. Therefore, the Markov assumption can be used to address this problem. It assumes that the probability of observing a feature \( x_i \) at a given position \( i \) of the sequence is only dependent on the features observed in the previous \((i - 1) \ldots i - c \) positions, and independent of the features observed in all of the positions before \( i - c \).

\[ p(x_i | x_1, \ldots, x_{i-1}) \equiv p(x_i | x_{i-c}, \ldots, x_{i-1}) \]
\[ p(x_n, \ldots, x_1) \equiv \prod_{i=1}^{n} p(x_i | x_{i-c}, \ldots, x_{i-1}) \]

Adhering to this, a Word2Vec model uses a sliding window \( S \) of size \( c \) over the corpus to learn word embeddings [42]. Within the window \( S \), the Continuous Bag-of-Words (CBOW) model of Word2Vec uses context (surrounding words) to predict a target word, and the skip-gram model of Word2Vec uses a target word to predict a context. For the CBOW model, Eq. 4 can be written as,

\[ \ln p(x_n, \ldots, x_1) \equiv \sum_{i=1}^{n} \ln p(x_i | x_{i-c}, \ldots, x_{i-1}) \]
\[ \equiv \sum_{i=1}^{n} \sum_{x_j \in S_i} \ln p(x_j | S_i - \{x_j\}) \]

where \( S_i \) represents the starting position of sliding window \( S \) at \( i \)th word (or feature). The objective of Word2Vec is to learn model parameters \( \theta \) (a.k.a. word embeddings or word vectors of \( N \) words) from a large unlabelled dataset to maximise \( p(x_N, \ldots, x_1) \) over the dataset. That is,
Regularising LSTM classifier by transfer learning for detecting…

\[
\arg\max_{\theta} \sum_{i=1}^{N-c+1} \sum_{s_j \in S_i} \ln p(x_j|S_j - \{x_j\}, \theta)
\]

The learned \( N \) word vectors (\( \theta \)) can be viewed as the approximate distributed representation of \( p(x_N, \ldots, x_1) \) over the dataset.

To learn the parameters of the classification model, the model is approximated to \( p(x) \), i.e. \( p(x_n, \ldots, x_1, \theta) \approx p(x_n, \ldots, x_1) = p(x) \).

It can be estimated as follows.

\[
p(C_k, x) = p(C_k|x)p(x) \approx p(C_k|x)p(x, \theta) \approx p(C_k|x, \theta)p(x, \theta) \quad (5)
\]

where \( \theta \) is the set of model parameters trained to approximate \( p(C_k|x) \). This process can be inferred as the following. Word vectors can be pretrained using a large unlabelled dataset and can be used to train a discriminative classifier model such as LSTM or CNN. For predicting unseen instances, probability \( p(C^*_k|x^*, C_k, x) \) can be used, where \( C^*_k \) is the class predicted of an unseen instance and \( x^* \) is the unseen instance vector. The probability of the class \( C^*_k \) of an instance \( x^* \) in the test dataset is conditioned on the instance itself and the training dataset.

As the training model can be regularised by pretraining, the performance of predicting test instances depends on the learned model parameters during training and pretraining, which can be seen from the joint probability distribution

\[
p(C^*_k, x^*, C_k, x) = p(C^*_k|x^*, C_k, x)p(x^*, C_k, x) \\
= p(C^*_k|x^*, C_k, x)p(x^*|C_k, x)p(C_k, x) \\
= p(C^*_k|x^*, C_k, x)p(x^*|C_k, x)p(C_k|x)p(x) \\
\approx p(C_k|x^*, C_k, x)p(x^*|C_k, x)p(C_k|x, \theta)p(x, \theta).
\]

It can be noted that the parameters \( \theta \) learned during pretraining with the unlabelled data and the parameters \( \theta \) learned during training with the labelled data can significantly impact the prediction performance of the classification model as the prediction probability \( p(C^*_k|x^*, C_k, x) \) is a part of the joint probability and the joint probability is approximated by \( \theta \) and \( \theta \).

Even though word embeddings have been used in a myriad of applications [42, 56], it ignores two important characteristics of a given sequence: (a) order of features in the sliding window \( S \) and (b) nonlinear interactions between features. Since Word2Vec uses a single hidden layer, it can capture only the linear interaction between features. However, features of a sequence (e.g. a sentence in a natural language) can have many levels of nonlinear interactions.

An effective language model (LM) should capture the order of features and their nonlinear interactions. It should encode the complexity of a language such as grammatical structure as well as distill a fair amount of knowledge from the corpus [25]. In this paper, we propose to achieve this by constructing a LM on multiple large datasets exhibiting distinct characteristic so it can learn useful knowledge encoded in those large unlabelled data [62]. To capture the order of features and their nonlinear interactions, we use neural network language model as described in the following section.
4.3 Neural network language model

A language model designed using NN to provide a probability distribution over sequences of words is called neural network language model (NNLM) [25]. Given a sequence of \( n \) words (i.e. features) \( \mathbf{x} = (x_1, \ldots, x_n) \) as input, it provides a probability \( p(x_1, \ldots, x_n) \) as output. A sequence-based deep neural network model (i.e. RNN and its variants (e.g. LSTM)) can represent a LM [25, 41]. A RNN-based model can preserve the order of features in a sequence and, by using multiple hidden layers, it can capture the nonlinear interactions between features. A simple RNN, as shown in Fig. 2, is constructed by repeatedly applying a function \( f_h \) that generates a hidden state \( \mathbf{h}_i \) for \( i \)th feature \( x_i \) represented with vector \( \mathbf{x}_i \), i.e.

\[
\mathbf{h}_i = f_h(\mathbf{x}_i, \mathbf{h}_{i-1}) = \varphi(\mathbf{x}_i \mathbf{W} + \mathbf{h}_{i-1} \mathbf{U} + \mathbf{b})
\]

where \( \mathbf{W} \) is the parameter (weight) matrix for input to hidden layer for \( i \)th feature, \( \mathbf{U} \) is the parameter matrix for hidden layer of \( (i - 1) \)th feature to hidden layer of \( i \)th feature, \( \mathbf{b} \) is bias vector and \( \varphi \) is a (nonlinear) activation function such as tanh or ReLU.

The hidden state is used to derive a vector of probabilities representing the network’s guess of the subsequent feature in the sequence. LSTM [22], a variant of RNN, uses a gating mechanism to ensure proper propagation of information through many steps of a sequence, to retain long-term dependencies. An NNLM aims to minimise the loss calculated based on the vector of probabilities and the actual next feature. In simple words, the context of all previous features in the sequence is encoded within the parameters \( \mathbf{W} \) of an RNN/LSTM and the probability of getting next word is distributed over the vocabulary using a Softmax function [25]. The model output \( \mathbf{o} \) can be defined as follows.

\[
\mathbf{o}_i = f_o(\mathbf{h}_i) = \sigma(\mathbf{h}_i \mathbf{V} + \mathbf{a})
\]

where \( \mathbf{V} \) is the parameter matrix for a hidden layer to output for \( i \)th feature, \( \mathbf{a} \) is the bias vector and \( \sigma \) is a softmax function used to convert the result into a probability distribution over vocabulary.

It can be noted that, similar to traditional LM, a RNN/LSTM-based NNLM can approximate joint probabilities over the feature sequences.
Regularising LSTM classifier by transfer learning for detecting…

\[ p(x) = p(x_1, \ldots, x_n) \approx \prod_{i=1}^{n} p(x_i|x_{i-1}, \theta) \quad (6) \]

4.4 Transfer learning for language model

A common TL approach is to learn \( p(x) \) from a single dataset [25, 38, 39]. When a model learns \( p(x) \) from a dataset \( D \), the learned probability distribution depends on the corpus \( D \). In other words, \( p(x) \) is conditioned on the corpus \( D \), i.e. \( p(x, D) = p(x|D)p(D) \). Since a LM built with a corpus can capture the knowledge of the corpus, using a large dataset that covers multiple related domains in TL can be useful. For example, for a LM to learn about computer programming, the model should also incorporate common knowledge on computer science. However, estimating \( p(x) \) using a LSTM-based LM (i.e. NNLM) model on a huge dataset that covers multitude of domains can be very expensive in terms of required computation and memory [8]. Additionally, it can learn irrelevant and misleading relationships in data due to interactions between different domains in a single corpus.

We propose an alternative approach based on transfer learning to incorporate knowledge gained from multiple datasets in a LSTM-based LM. Let there be \( m \) number of corpora from which the knowledge is gained.

\[ p(x, D_1, \ldots, D_m) = p(x|D_1, \ldots, D_m)p(D_1, \ldots, D_m) = p(x|D_1, \ldots, D_m)p(D_1|D_2, \ldots, D_m) \cdots p(D_{m-1}|D_m)p(D_m) \]

If we assume the datasets are independent of each other, applying the Naive Bayes assumption, we can write

\[ p(x|D_1, \ldots, D_m)p(D_1|D_2, \ldots, D_m) \cdots p(D_{m-1}|D_m)p(D_m) \propto p(x|D_1, \ldots, D_m)p(D_1)p(D_2) \cdots p(D_m) \]

\[ = p(x|D_1)p(D_1)p(x|D_2)p(D_2) \cdots p(x|D_m)p(D_m) \]

\[ = \prod_{i=1}^{m} p(x|D_i)p(D_i) \]

A LSTM model built on corpus \( D_i \) to learn \( p(x|D_i)p(D_i) \) will have its parameters \( \omega_i \). It can be expressed as follows.

\[ \prod_{i=1}^{m} p(x|D_i)p(D_i) \approx \prod_{i=1}^{m} p(x|D_i, \omega_i)p(D_i, \omega_i) \]

\[ = \prod_{i=1}^{m} p(x|D_i, \omega_i)p(\omega_i|D_i)p(D_i) \]

If the same LM model is sequentially built from the given \( m \) datasets, parameters \( \omega_i \) learned on \( i^{th} \) dataset will only depend on the parameters \( \omega_{i-1} \) learned on the \((i - 1)^{th}\) dataset, applying the Markov assumption.
\[
\prod_{i=1}^{m} p(x|D_i, \omega_i)p(\omega_i|D_i)p(D_i) \approx \prod_{i=1}^{m} p(x|D_i, \omega_i)p(\omega_i|D_i, \omega_{i-1})p(D_i)
\]

Here \(\omega_0\) is the initial weight that might be assigned randomly. Assuming the same probability (or uncertainty) for each dataset, transfer learning can be expressed as follows.

\[
p(x, D_1, \ldots, D_m) \approx \prod_{i=1}^{m} p(x|D_i, \omega_i)p(\omega_i|D_i, \omega_{i-1})p(D_i)
\]

\[
= \prod_{i=1}^{m} p(x|D_i, \omega_i)p(\omega_i|D_i, \omega_{i-1})
\]

\[
\propto \sum_{i=1}^{m} \ln\left(p(x|D_i, \omega_i)p(\omega_i|D_i, \omega_{i-1})\right)
\]

Following can be inferred from Eq. 7. (1) Each dataset \(D_i\) relevant to the application domain of LM can reduce uncertainty. This reinforces the previous findings based on word embedding that if word vectors of Word2Vec are pretrained using a corpus relevant to the target task domain, performance of the target task is significantly improved [5]. (2) Pre-training of LSTM-LM should be done by the order of the dataset of general population distribution because the parameter vector \(\omega_i\) depends on \(\omega_{i-1}\). For example, we can approximate the population distribution of Queensland (i.e. specific) from that of Australia (i.e. general) but the opposite is not true.

### 4.5 Classifier regularised by LSTM-based LM

For the downstream task of classification, a LSTM is trained to learn \(p(C_k|x, \theta)p(x, \theta) \approx p(C_k, x)\) (Eq. 5) on a small training dataset. The model \(p(C_k^*|x^*, C_k, x)\) can be used to predict the class of an unseen instance. For example, the probability over the feature sequences \(p(x, \theta) \approx p(x, D_1, D_2, D_3)\) is learned by applying the pretraining NNLM on three datasets \(D_1, D_2\) and \(D_3\).

Figure 3 shows the process of transfer learning through LSTM-based LM to a LSTM classification model. Layers 1 to 3 are stacked LSTM layers. The LSTM-based LM (the left hand side model of Fig. 3) is generated using three staked layers along with embedding layer and LM softmax layer. The LM softmax layer is active during pretraining of LM with a sequence of unlabelled datasets and then it is freezed.

As shown by the right hand side model of Fig. 3, the class softmax layer and linear layer are augmented after the LM is pretrained. These two layers along with the pretrained LM active layers are trained with the small labelled dataset to learn the classification task of misogynistic tweets. The main task of these additional two layers is to learn \(p(C_k|x, \theta)\). The combined network learns \(p(C_k|x, \theta)p(x, \theta)\). The additional two layers are augmented at the end of NNLM in this model to assure that \(\theta\) is learned from the fine-tuning of \(\theta\) and \(D_t\) during classification training. We call the process of building the classification model regularised by transfer learning, \(p(C_k|x, \theta)p(x, \theta)p(\theta|\theta, D_t) \approx p(C_k, x)\), as LSTM-based LM regularised classifier (LSTM-L).
Empirical evaluation

The primary objectives of experiments are to show: (a) the effectiveness of transfer learning in regularising overfitting (or uncertainty) when there is a small set of labelled data available for training the classifier; (b) the effectiveness of LSTM-based LM over Word2Vec as the uncertainty regularisation technique; and (c) the classification performance improvement by training a LM on datasets from general to specific domain. All these experiments were conducted in order to achieve the best accuracy performance in detecting misogynistic abusive tweets.

For training the proposed LSTM-L classification model, we use concat pooling and gradual unfreezing techniques as in [24]. For building the LM, we model NNLM using the state-of-the-art AWD-LSTM [39] which is a standard LSTM with various tuned dropout hyper parameters. The architecture and hyper parameters of NNLM is the same as that used in [24]. We use ReLU activations for the intermediate layers and the softmax activation at the last layer that outputs probability distributions over the target vocabulary (for LM) or classes (for classification). Hyperparameters are tuned using cross-validation. We used Python Machine Learning Library PyTorch\(^1\) to implement this model. Coding was done using Jupyter Notebook\(^2\) and executed on a Linux machine. High performance computing facilities used in this research were provided by eResearch Office, Queensland University of Technology, Brisbane, Australia.

---

\(^1\) [https://pytorch.org/](https://pytorch.org/).
\(^2\) [https://jupyter.org/](https://jupyter.org/).
5.1 Data collection

A number of datasets are used in experiments. Particularly, two small labelled datasets are used for training classifiers and three large unlabelled datasets are used for pretraining the NNLM (i.e. LSTM-based LM). A description of these datasets are given below.

5.1.1 QMI Dataset for misogynistic tweet detection

We collected tweets using the Twitter’s streaming API. A set of tweets that contain any of the three main misogynistic keywords (i.e. whore, slut, rape) [4] was collected by a random sampling during the period of January 2007–March 2018 in Australian domain. After removing tweets that contain a lot of non-English words, a total of 5000 tweets was obtained. Following the misogynistic tweet definition in Sect. 3, we manually labelled the 5000 tweets where 1800 (36%) tweets were labelled misogynistic and 3200 (64%) were labelled non-misogynistic. We call this data as QUT Misogyny Identification (QMI) data.

We used 80-20% as training and testing subsets. We used ten-fold cross-validation to tune hyperparameters and used the Porter’s suffix-stripping algorithm and invalid punctuation character fixing for preprocessing.

Note, our objective in this paper is to demonstrate the technical capability of machine learning models to handle the microblogging data in general, instead of a preprocessed data. We did not use the advanced preprocessing steps such as spelling correction, elongated word normalisation (‘gooooood’ becomes ‘good’), word segmentation on hashtags (‘#womensuck’ becomes ‘women suck’), and unpacking contractions (e.g. ‘can’t’ becomes ‘can not’), as detailed in [63]).

The advanced preprocessing may improve the prediction accuracy; however, it is subjected to the dataset and the model in hand. Different preprocessing is effective in different datasets and models. For example, a model that can capture word sequence (e.g. LSTM) might be benefited when ‘#womensuck’ is segmented into ‘women suck’, while a bag-of-words-based model can be benefited when ‘#womensuck’ is kept unsegmented because ‘#womensuck’ can be a discriminative word for misogynistic tweets. It becomes difficult to discern the capability of a machine learning model or the capability of applying advanced preprocessing on the dataset for achieving desired accuracy.

5.1.2 AMI dataset for misogynistic tweet detection

To see how effectively the classification models work on other misogyny detection datasets, we have also used the automatic misogyny identification (AMI) English dataset used in recent competitions [16]. The training and testing sets contain 4000 and 1000 labelled tweets, respectively. The training set has 1785 (45%) misogynous tweets and 2215 (55%) non-misogynous tweets, while the testing set has 460 (46%) misogynous tweets and 540 (54%) non-misogynous tweets.
5.1.3 Pretraining datasets for NNLM

We use the following three corpora for sequentially pretraining LSTM-based LM (i.e. NNLM).

\( D_1 \): The goal of using this corpus is to capture general properties of the English language. We pretrain the NNML model on Wikitext-103 that contains 28,595 preprocessed Wikipedia articles and 103 million words [40]. After pretraining on \( D_1 \), we approximate the probability distribution \( p(x|D_1, \omega_1) \).

\( D_2 \): The goal of using this corpus is to bridge the data distribution between the target task domain (i.e. abusive tweets) and the general domain (i.e. standard language). This is because the target task is likely to come from a different distribution than the general corpus. \( D_2 \) should be chosen such that it has commonalities with both \( D_1 \) reflecting a general domain (Wikipedia) and the corpus \( D_3 \) reflecting a target domain (tweets).

We use 25k reviews from movie review dataset IMDb [35] without labels as \( D_2 \). Documents in IMDb are generally a few paragraphs long expressed by people. Similar to Wikipedia, they discuss topics, stories or personnel with some details and they are noisy. Similar to tweets, they are written by general people and not peer reviewed. More importantly, both IMDb reviews and tweets often express personal opinions and sarcasms.

As \( p(x|D_2, \omega_2)p(\omega_2|D_2, \omega_1) \) is approximated on \( D_2 \), \( \omega_2 \) can be considered tuned with \( D_2 \) and \( \omega_1 \). \( D_2 \) may not completely be subset of \( D_1 \), i.e. \( D_2 \) may contain some exclusive information other than \( D_1 \). This means \( p(x|D_2, \omega_2)p(\omega_2|D_2, \omega_1) \) is more specific and less uncertain than \( p(x|D_1, \omega_1) \). We propose using discriminative fine-tuning (tune each layer of LSTM with different learning rates) and slanted triangular learning rates (first rapidly increases the learning rate and then slowly decays) [24] for fine tuning the NNLM model with \( D_2 \).

\( D_3 \): The goal of this corpus is to capture the domain specific language properties of the target task. We tune NNML on a dataset of about 1.9 million random tweets. Once fine-tuning of NNLM on \( D_3 \) is done, we get an approximation for \( p(x,D_1,D_2,D_3) \approx p(x,\omega) \) which is an approximation for \( p(x) \). Similar to \( D_2 \), we use discriminative fine-tuning and slanted triangular learning rates for fine tuning the NNLM model with \( D_3 \).

5.2 Evaluation measures

We used six standard classification evaluation measures: Accuracy (Ac), Precision (Pr), Recall (Re), \( F_1 \) Score (\( F_1 \)), Cohen Kappa (CK) and Area Under Curve (AUC). We also report True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN) values. A description of evaluation measures is given in “Appendix A”.

5.3 Baseline models

We have implemented 12 baseline models to compare the performance of our proposed LSTM-L.

- Word2Vec regularised baselines include (1) LSTM regularised by Word2Vec (LSTM-W) [22] and (2) CNN regularised by Word2Vec (CNN-W) [5]. LSTM-W
has 100 units, 50% dropout, binary cross-entropy loss function, Adam optimiser and sigmoid activation. The hyperparameters of CNN-W are set as in [5]. Word vectors have 200 dimensions and are pretrained on 0.64 billion random tweets. A Continuous Bag-of-Words Word2vec [42] model is used in pretraining while minimum count for word is set to 100.

- LSTM without TL (LSTM-P) [22]. This is a traditional LSTM model that has not been pretrained by any data. Similar to LSTM-W, LSTM-P has 100 units, 50% dropout, binary cross-entropy loss function, Adam optimiser and sigmoid activation.

- Combined CNN and GRU (CNN+GRU) [63]. This model combines a CNN and a Gated recurrent units (GRU) layer. The GRU layer takes input from the maxpooling layer of CNN. GRU is similar to LSTM but GRU has simpler structure and less parameters to train, which results in faster training. For this model, we use the same hyper parameter as used in [63].

- Feedforward deep neural network (DNN) [20]. It has five hidden layers, each layer containing eighty units, 50% dropout applied to the input layer and the first two hidden layers, softmax activation and 0.04 learning rate. For all neural network-based models, hyperparameters are manually tuned based on cross-validation.

- Non-NN models including support vector machines (SVM) [21] linear SVM (SVM-L) and nonlinear SVM (SVM-N), random forest (RF) [32], XGBoost (XGB) [9], multinomial Naive Bayes (MNB) [29], k-nearest neighbours (kNN) [59] and ridge classifier (RC) [23]. Hyperparameters of all these models are automatically tuned using ten-fold cross-validation and GridSearch using sklearn library.

None of the models, except LSTM-L, LSTM-W and CNN-W, are pretrained or utilised unlabelled dataset.

### Table 2 Performances of classification models on the QMI dataset

| Model   | TP  | TN  | FP  | FN  | Ac  | Pr  | Re  | F1  | CK  | AUC |
|---------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| LSTM-L  | 274 | 575 | 66  | 88  | 0.846 | 0.806 | 0.757 | 0.781 | 0.663 | 0.827 |
| LSTM-W  | 173 | 585 | 56  | 189 | 0.756 | 0.755 | 0.478 | 0.585 | 0.424 | 0.695 |
| LSTM    | 167 | 561 | 80  | 195 | 0.726 | 0.676 | 0.461 | 0.548 | 0.362 | 0.668 |
| CNN-W   | 255 | 530 | 111 | 107 | 0.783 | 0.697 | 0.704 | 0.701 | 0.530 | 0.766 |
| CNN+GRU| 73  | 590 | 51  | 289 | 0.661 | 0.589 | 0.202 | 0.300 | 0.142 | 0.561 |
| DNN     | 195 | 509 | 132 | 167 | 0.702 | 0.753 | 0.794 | 0.773 | 0.340 | 0.666 |
| SVM-L   | 195 | 541 | 100 | 167 | 0.734 | 0.661 | 0.539 | 0.594 | 0.399 | 0.691 |
| SVM-N   | 230 | 536 | 105 | 132 | 0.764 | 0.687 | 0.635 | 0.660 | 0.479 | 0.736 |
| RF      | 153 | 596 | 45  | 209 | 0.747 | 0.773 | 0.423 | 0.546 | 0.391 | 0.676 |
| XGB     | 185 | 555 | 86  | 177 | 0.738 | 0.683 | 0.511 | 0.585 | 0.399 | 0.688 |
| MNB     | 170 | 595 | 46  | 192 | 0.763 | 0.787 | 0.470 | 0.588 | 0.436 | 0.699 |
| kNN     | 25  | 622 | 19  | 337 | 0.645 | 0.568 | 0.069 | 0.123 | 0.049 | 0.520 |
| RC      | 207 | 541 | 100 | 155 | 0.746 | 0.674 | 0.572 | 0.619 | 0.430 | 0.708 |

Boldfaced numbers indicate best values
5.4 Effectiveness of LSTM-LM in regularising overfitting

5.4.1 Classifier models comparison: performance on the QMI dataset

The classification performance of models on the held-out test-subset (20%) of QMI dataset is summarised in Table 2. Based on the results, following observations can be made.

(a) Performance of LSTM-L is significantly better than all the baseline models. Improvements in accuracy, precision, recall, Cohen Kappa score and AUC of LSTM-L over CNN-W (which is the second best performing model) are 8.153%, 15.668%, 7.451%, 11.431%, 25.044% and 8.012%, respectively. It ascertains that the LSTM model fine-tuned on top of LSTM-based LM is an effective way to detect misogynistic tweets where there is a small labelled dataset available.

(b) Performance of LSTM-L (regularised by a LSTM-based LM) is significantly better than that of LSTM-W and CNN-W (regularised by Word2Vec). This complements the theoretical finding in Sect. 4.2 that showed a LSTM-based model can approximate LM better than Word2Vec. It ascertains that a better LM approximation can result in better classifier model regularisation.

(c) Results show that the regularisation of a NN model by underlying data distribution can complement the regularisation of NN architecture (e.g. random dropout). Performance of LSTM-L is superior to LSTM-W and the performance of LSTM-W is better than LSTM-P and DNN. An NN model such as LSTM-P or DNN is commonly regularised by random dropout, i.e. weights of some nodes are randomly assigned to zero. This reduces overfitting [49] when the labelled dataset is medium. On the other hand, weights of embedding layer in LSTM-W is initialised from pretrained word vectors, and the weights in embedding layer and hidden layers (except the last linear layer) in LSTM-L are initialised by a pretrained LM on unlabelled datasets. In other words, LSTM-W is regularised by underlying data distribution of word semantics and LSTM-L is regularised by underlying data distribution of language, in addition to random dropout. The empirical results and theoretical analysis (Sect. 4.1) reinforce this.

(d) LSTM-based models learn sequences (i.e. order of features) in the text data. Consequently, they need more training data compared with other NN models such as CNN and DNN; otherwise, a LSTM model overfits. It can be seen by the lower recall values of LSTM-W and LSTM-P in comparison with CNN-W and DNN as they show overfitting the training data. However, LSTM-L can learn sequences in a domain even with small training set by utilising knowledge gained from pretraining datasets. It can reduce overfitting as seen by the higher or equal recall value than CNN-W, LSTM-W, LSTM-P and DNN.

(e) Performance of all NN models augmented by transfer learning (simple or complex) is better than other models. This confirms that the regularisation of classifiers by transfer learning (interpreted as regularisation by underlying data distribution in this paper) can reduce overfitting problem.

(f) We intentionally did not stratified the labelled dataset, i.e. there were more negative instances (64%) than positive instances (36%). As a result, traditional models are being biased to the negative instances. For example, kNN could not detect enough misogynistic tweets (true positive is 25 out of a total 362), even though it scores the highest number of true negative (622 instances out of a total 641) and the lowest number of false negative (19 instance). LSTM-W also performed poorly in detecting true posi-
tives (173), even though it performed well for true negatives (585). On the other hand, LSTM-L could detect highest number of misogynistic tweets (true positive is 274). LSTM-based LM is able to regularise distribution bias in labelled dataset. This is an important achievement as having the number of negative examples higher is a common scenario in many datasets in practice.

5.4.2 Classifier models comparison: performance on the AMI dataset

The performance of LSTM-L and all baseline models on another misogyny dataset is shown in Table 3. Since the AMI dataset was used in Evalita 2018 Task of automatic
misogyny identification [16], we also present the results of contributing teams in Table 4. In Evalita 2018, the best performing team hateminers used the logistic regression method and developed their models based on vector representation that concatenates sentence embedding, TF-IDF and average word embeddings [16]. The second best team resham used several external domain-related knowledge such as links, hashtags, emojis, swear words, sexist slurs and women-related words in their model [16]. The third best team bakarov used TF-IDF coupled with singular value decomposition and a boosting decision tree classifier [16]. The team StopPropagHate that used a simple dense neural network was amongst the worse performers.

Comparison of Accuracy result of Tables 3 and 4 shows that LSTM-L outperforms the best team, LSTM-W equals the third best result and MNB outperforms the fourth best result. As expected, linear SVM (SVM-L) gives the similar result as AMI-BASELINE because AMI-BASELINE uses the linear SVM model. Whereas SVM-N outperforms AMI-BASELINE as SVM-N uses kernel trick. Similar to results on QMI (Table 2), comparison of the models in Table 3 shows that overall the best performing model is LSTM-L, the second best performing model is LSTM-W. All these results show that regularising an LSTM classifier by transfer learning result in an improved classification performance.

5.4.3 Performance on random tweets

To see how the trained LSTM-L performs when the test tweets do not have misogynistic keywords in them, we have tested it on a subset of randomly collected tweets that do not contain one of those three misogynistic keyword. The experimental results show that the model can detect some of such misogynistic tweets. Some examples are shown in Table 5. However, in such a setting, a higher number of False Positive was observed. A possible reason is that LSTM-L is trained on tweets with a different data distribution than these tweets. We also observed that by adding additional non-misogynistic tweets (which is easy to obtain) into the training set, this higher False Positive effect can be reduced. We will do a rigorous investigation in this direction in our future research.

5.4.4 Impact of pretraining datasets

We conducted experiments with the QMI dataset to see the effects of different combinations of datasets used to pretrain LSTM-based LM. Results in Table 6 show that the
The performance of LSTM-L is best when the language model is built with $D_1 > D_2 > D_3$ (general to specific). Its performance is better than $D_3 > D_2 > D_1$ (specific to general), $D_3 > D_2$ (partial specific to general) and $D_3$ (specific) or $D_1$ (general). It ascertains that pretraining LM with multiple datasets from general to specific domain is effective. The next best performance is obtained with $D_3$, that is better than $D_1$. It indicates that knowledge captured by LM from a dataset specific to problem domain is more useful than the knowledge captured from a general domain dataset. The performance with $D_3$ is better than $D_3 > D_2 > D_1$ and $D_3 > D_2$. This shows that going from specific to general domain always harms the performance.

The proposed TL obtains $p(C_k, x)$ by multiplying regularisation $p(x)$ with $p(C_k|x)$. Consequently, a dataset in a given domain tries to put a feature sequence to its context. The general domain $D_1$ is a subset of Wikipedia that has more non-abusive distribution of features. As a result, $D_1$ gives a higher true negative score of 540 compared with 531 score of $D_3$. On the other hand, $D_3$ is a subset of random tweets that has more distribution of abusive features when compared with Wikipedia. As a result, $D_3$ gives a higher true positive score of 298 when compared with 262 score of $D_1$. With the addition of each specific domain dataset, true positive score increases. LSTM-L is a binary classifier so improved positive identification can result in improved negative identification. However, when seen individually, true negative score in $D_1$ is higher than that in $D_3$.

These results indicate that the domain of pretraining datasets and their order can affect the LM performance.

### Table 6  Impact of datasets on pretraining LSTM-based LM

|                | $D_1$ | $D_3$ | $D_1 > D_2$ | $D_3 > D_2$ | $D_1 > D_2 > D_3$ | $D_3 > D_2 > D_1$ |
|----------------|-------|-------|-------------|-------------|-------------------|-------------------|
| True Positive  | 262   | 298   | 271         | 301         | 274               | 262               |
| True Negative  | 540   | 531   | 547         | 504         | 575               | 553               |
| False Positive | 101   | 110   | 94          | 137         | 66                | 88                |
| False Negative | 100   | 64    | 91          | 61          | 88                | 100               |
| Accuracy       | 0.800 | 0.827 | 0.816       | 0.803       | **0.846**         | 0.813             |
| Precision      | 0.722 | 0.730 | 0.742       | 0.687       | **0.806**         | 0.749             |
| Recall         | 0.724 | 0.823 | 0.749       | **0.831**   | 0.757             | 0.724             |
| F1 Score       | 0.723 | 0.774 | 0.746       | 0.753       | **0.781**         | 0.736             |
| Cohen Kappa    | 0.566 | 0.634 | 0.601       | 0.591       | **0.663**         | 0.591             |
| AUC            | 0.783 | 0.826 | 0.801       | 0.809       | **0.827**         | 0.793             |

$D_1$: LM pretrained on dataset $D_1$ (Wikipedia—most generic) only; $D_1 > D_2 > D_3$: LM pretrained on $D_1$ first then $D_2$ (IMDb) and then $D_3$ (random tweets) (generic-to-specific)

Boldfaced numbers indicate best values

5.4.5 Data augmentation performances

To compare the LSTM-based LM transfer learning with data augmentation, experiments were conducted with multiple practices of augmentation/expansion on QMI. The goal is to evaluate if the performance of other LSTM-based models (e.g. LSTM-W) with data augmentation/expansion can be improved to that of LSTM-L. We used two processes to
generate augmented data: (1) word vectors pretrained on unlabelled tweets and (2) topics identified by Nonnegative matrix factorisation (NMF) [28] on the labelled tweets (QMI).

A total of six augmentation techniques were performed. AT1: Words in a labelled tweet are randomly replaced by semantically similar words from the word vector space to create a new tweet. AT2: Discriminative Words in a labelled tweet are randomly replaced by semantically similar words from word vector space to create a new tweet. A discriminative word is a word that appears more frequently in tweets of a specific class. AT3: A tweet is expanded by adding its semantically similar words found from word vector space to create a new tweet. Unlike AT1, words are not replaced. AT4: A tweet is expanded by adding its semantically similar words found from NMF to create a new tweet. AT5: Use the words of topics relevant to a tweet as a new tweet. Topics are discovered by NMF. AT6: A set of words from word vector space that is semantically similar to a tweet is used as a new tweet.

Each new tweet is assigned with the label of the tweet that was used to create it. Table 7 reports the performance of LSTM-W model trained with each augmentation technique. The LSTM-W model trained without any augmentation was labelled as AT0. Results show that these ways of augmentation do not improve the accuracy. A potential reason behind is the small size of each tweet, presence of noise and the lower co-occurrences of word pairs. We conjecture that additional external features (i.e. words) may distort the sequences and patterns that exist in the original tweets. Since the LSTM-W classifier largely depends on learning these sequences and patterns, the performance degrades.

This implies that capturing internal sequences and patterns effectively is very important for the performance of NN-based text classification models, especially LSTM-based models. As pretrained LSTM-based LM can capture the sequences and patterns in a language from unlabelled datasets, LSTM-based LM can significantly improve misogynistic tweet detection performance, while the data augmentation/expansion (especially policies used in this paper) fail.

### 5.4.6 Discussion

The best performance obtained by the proposed model LSTM-L is about 85% and 73% for QMI and AMI datasets respectively. This may be limited due to the nature of the labelled data. For example, the tweet labelling method of these datasets has the following limitations: (a) The labelling is based on a literal interpretation of the text; with limited context (e.g. no information about user behaviour), we are likely to label some sarcasm or humour.

| Table 7 Performance of LSTM-W with data augmentation policies |
|-----------------|---|---|---|---|---|---|
|                | AT0 | AT1 | AT2 | AT3 | AT4 | AT5 | AT6 |
| True positive  | 173 | 126 | 142 | 145 | 133 | 134 | 148 |
| True negative  | 585 | 594 | 596 | 581 | 577 | 596 | 591 |
| False positive | 56  | 47  | 45  | 60  | 64  | 45  | 50  |
| False negative | 189 | 236 | 220 | 217 | 229 | 228 | 214 |
| Accuracy       | 0.756 | 0.72 | 0.74 | 0.72 | 0.71 | 0.73 | 0.74 |
| Precision      | 0.755 | 0.73 | 0.76 | 0.71 | 0.68 | 0.75 | 0.75 |
| Recall         | 0.478 | 0.35 | 0.39 | 0.4  | 0.37 | 0.37 | 0.41 |
| F1 score       | 0.585 | 0.47 | 0.52 | 0.51 | 0.48 | 0.5  | 0.53 |
| Cohen kappa    | 0.424 | 0.31 | 0.36 | 0.34 | 0.3  | 0.34 | 0.37 |
| AUC            | 0.695 | 0.64 | 0.66 | 0.65 | 0.63 | 0.65 | 0.67 |
as misogyny. (b) We are only labelling tweets written in English. (c) Filtering the tweets by keywords only, we may not catch abuse that appears to be ordinary misogyny, e.g. *get back in the kitchen*. (d) Filtering the tweets by keywords only, we may not identify harassment that is targeted and organised harassment, either ongoing over time or involving many participants, but does not use one of our keywords.

6 Conclusion

This paper presented a novel method of NN-based transfer learning to regularise a NN classifier when only a small amount of labelled data is available. It gave a Bayesian interpretation to transfer learning and showed that transfer learning can be viewed as an uncertainty regularisation process in Bayesian inference. It pretrained an LSTM-based LM using a sequence of three datasets from general to the task-specific domain and used the pretrained LM to regularise a LSTM classifier. Though it evaluated the proposed method on the problem of misogynistic tweet detection, the method is applicable to any other situation where it is difficult to get a large labelled text dataset.

A series of experiments were conducted to investigate its effectiveness. The regularised LSTM classifier performs significantly better than other state-of-the-art models. Theoretical analysis and experimental results show that the regularisation of uncertainty can significantly impact the accuracy of an LSTM classifier. Specifically, the LSTM-based LM regularisation significantly improved classification accuracy when compared with the Word2Vec regularisation that captures more uncertainty over data distribution.

By viewing transfer learning as a regularisation technique for text classifiers, this paper implies the potential of transfer learning through other models (e.g. Gaussian Distribution, Convolutional Neural Network, etc.). Also, this implies the use of transfer learning through feature selection from unlabelled dataset, e.g. selecting relevant features from the unlabelled datasets that can regularise the semantics of available and missing features in a small labelled dataset. Bayesian interpretation may be useful to analysis and estimate uncertainties in transfer learning to choose the right datasets and models for pretraining. This will be our future direction of research.

Acknowledgements his research was partially supported by the QUT IFE Catapult fund. Suzor is the recipient of an Australian Research Council DECRA Fellowship (project number DE160101542).

Appendix A: Description of evaluation measures

- True Positive (TP): True positives are instances classified as positive by the model that actually are positive.
- True Negative (TN): True negatives are instances the model classifies as negative that actually are negative.
- False Positive (FP): False positives are instances identified by model as positive that actually are negative.
- False Negative (FN): False negatives are instances the model classifies as negative that actually are positive.
- Accuracy (Ac): It is the percentage of correctly classified instances, and it is calculated as $\frac{TP + TN}{TP + TN + FP + FN}$. 
– Precision (Pr): It calculates a model’s ability to return only relevant instances. It is calculated as \( \frac{TP}{TP+FP} \).
– Recall (Re): It calculates a model’s ability to identify all relevant instances. It is calculated as \( \frac{TP}{TP+FN} \).
– \( F_1 \) Score (\( F_1 \)): A single metric that combines recall and precision using the harmonic mean. \( F_1 \) Score is calculated as \( 2 \times \frac{\text{precision}}{\text{precision} + \text{recall}} \).
– Cohen Kappa (CK): Cohen’s kappa score is used to measure inter-rater and intra-rater reliability for categorical items [37]. It is calculated as \( \frac{OA-AC}{1-AC} \), where OA is the relative observed agreement between predicted labels and actual labels and AC is the probability of agreement by chance.
– Area Under Curve (AUC): Area under the receiver operating characteristic (ROC) curve is called area under the curve (AUC). ROC plots the true positive rate versus the false positive rate as a function of the model’s threshold for classifying a positive. AUC calculates the overall performance of a classification model.

References

1. Ahluwalia R, Soni H, Callow E, Nascimento A, De Cock M (2018) Detecting hate speech against women in English tweets. EVALITA Eval NLP Speech Tools Ital 12:194
2. Amnesty International (2018) Toxic twitter—a toxic place for women. https://bit.ly/2FZYQhV
3. Badjatiya P, Gupta S, Gupta M, Varma V (2017) Deep learning for hate speech detection in tweets. In: Proceedings of the 26th international conference on world wide web companion, international world wide web conferences steering committee, pp 759–760
4. Bartlett J, Norrie R, Patel S, Rumpel R, Wilberley S (2014) Misogyny on twitter. Demos. Retrieved from analysis and policy observatory website https://apo.org.au/node/39610
5. Bashar MA, Nayak R, Suzor N, Weir B (2018) Misogynistic tweet detection: modelling CNN with small datasets. In: The 16th Australasian data mining conference
6. Blundell C, Cornebise J, Kavukcuoglu K, Wierstra D (2015) Weight uncertainty in neural network. In: International conference on machine learning, pp 1613–1622
7. Bouchard G, Triggs B (2004) The tradeoff between generative and discriminative classifiers. In: 16th IASC international symposium on computational statistics (COMSTAT’04), pp 721–728
8. Bradbury J, Merity S, Xiong C, Socher R (2016) Quasi-recurrent neural networks. arXiv preprint arXiv:1611.01576
9. Chen T, Guestrin C (2016) Xgboost: a scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. ACM, pp 785–794
10. Dai AM, Le QV (2015) Semi-supervised sequence learning. In: Advances in neural information processing systems, pp 3079–3087
11. Davidson T, Warmsley D, Macy M, Weber I (2017) Automated hate speech detection and the problem of offensive language. arXiv preprint arXiv:1703.04009
12. Devlin J, Chang M-W, Lee K, Toutanova K (2018) Bert: pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805
13. Downey A (2012) Think Bayes: Bayesian statistics made simple. Green Tea Press, Needham
14. Dragiewicz M, Burgess J, Matamoros-Fernández A, Salter M, Suzor NP, Woodlock D, Harris B (2018) Technology facilitated coercive control: domestic violence and the competing roles of digital media platforms. Fem Med Stud 18:1–17
15. Fadaee M, Bisazza A, Monz C (2017) Data augmentation for low-resource neural machine translation. In: Proceedings of the 55th annual meeting of the association for computational linguistics (volume 2: short papers), vol 2, pp 567–573
16. Fersini E, Nozza D, Rosso P (2018) Overview of the evalita 2018 task on automatic misogyny identification (AMI). In: Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA’18). Italy, Turin
17. Gal Y (2016) Uncertainty in deep learning. University of Cambridge, Cambridge
18. 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
19. Gitari ND, Zuiping Z, Damien H, Long J (2015) A lexicon-based approach for hate speech detection. Int J Multimed Ubiquitous Eng 10(4):215–230
20. Glorot X, Bengio Y (2010) Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the thirteenth international conference on artificial intelligence and statistics, pp 249–256
21. Hearst MA, Dumais ST, Osnana E, Platt J, Scholkopf B (1998) Support vector machines. IEEE Intell Syst Appl 13(4):18–28
22. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780
23. Hoerl AE, Kennard RW (1970) Ridge regression: applications to nonorthogonal problems. Technometrics 12(1):69–82
24. Howard J, Ruder S (2018) Universal language model fine-tuning for text classification. In: Proceedings of the 56th annual meeting of the association for computational linguistics (volume 1: long papers), vol 1, pp 328–339
25. Jojzeiowicz R, Vinyals O, Schuster M, Shazeer N, Wu Y (2016) Exploring the limits of language modeling. arXiv preprint arXiv:1602.02410
26. Kim Y (2014) Convolutional neural networks for sentence classification. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pp 1746–1751
27. Kwok I, Wang Y (2013) Locate the hate: detecting tweets against blacks. In: Twenty-seventh AAAI conference on artificial intelligence
28. Lee DD, Seung HS (2001) Algorithms for non-negative matrix factorization. In: Advances in neural information processing systems, pp 556–562
29. Lewis DD (1998) Naive (Bayes) at forty: the independence assumption in information retrieval. In: European conference on machine learning. Springer, pp 4–15
30. Li Y, Alzarni A, Zhong N (2010) Mining positive and negative patterns for relevance feature discovery. In: Proceedings of the 16th ACM SIGKDD international conference on knowledge discovery and data mining. ACM, Washington, pp 753–762
31. Li Y, Gal Y (2017) Dropout inference in bayesian neural networks with alpha-divergences. In: Proceedings of the 34th international conference on machine learning. JMLR.org, vol 70, pp 2052–2061
32. Liaw A, Wiener M et al (2002) Classification and regression by randomforest. R News 2(3):18–22
33. Liu P, Li W, Zou L (2019) Nuli at semeval-2019 task 6: transfer learning for offensive language detection using bidirectional transformers. In: Proceedings of the 13th international workshop on semantic evaluation, pp 87–91
34. Logeswara L, Lee H (2018) An efficient framework for learning sentence representations. arXiv preprint arXiv:1803.02893
35. Maas AL, Daly RE, Pham PT, Huang D, Ng AY, Potts C (2011) Learning word vectors for sentiment analysis. In: Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies. Association for Computational Linguistics, vol 1, pp 142–150
36. MacKay DJ (1992) A practical bayesian framework for backpropagation networks. Neural Comput 4(3):448–472
37. McHugh ML (2012) Interrater reliability: the kappa statistic. Biochem Med 22(3):276–282
38. Melis G, Dyer C, Blunsom P (2017) On the state of the art of evaluation in neural language models. arXiv preprint arXiv:1707.05589
39. Merity S, Keskar NS, Socher R (2017) Regularizing and optimizing LSTM language models. arXiv preprint arXiv:1708.02182
40. Merity S, Xiong C, Bradbury J, Socher R (2016) Pointer sentinel mixture models. arXiv preprint arXiv:1609.07843
41. Mikolov T, Karafiát M, Burget L, Černocký J Khudanpur S (2010) Recurrent neural network based language model. In: Eleventh annual conference of the international speech communication association
42. 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
43. Molina-González MD, Plaza-del Arco FM, Martín-Valdivia M, Ureña López L. (2019) Ensemble learning to detect aggressiveness in mexican spanish tweets. In: Proceedings of the first workshop for Iberian languages evaluation forum (IberLEF 2019), CEUR WS proceedings
44. Pittilis GK, Ramampiaro H, Langseth H (2018) Detecting offensive language in tweets using deep learning. arXiv preprint arXiv:1801.04433
45. Radford A, Narasimhan K, Salimans T, Sutskever I (2018) Improving language understanding with unsupervised learning. Technical report, OpenAI.
46. Rumelhart DE, Hinton GE, Williams RJ (1986) Learning representations by back-propagating errors. Nature 323(6088):533.
47. Sharif Razavian A, Azizpour H, Sullivan J, Carlsson S (2014) CNN features off-the-shelf: an astounding baseline for recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition workshops, pp 806–813.
48. Silva L, Mondal M, Correa D, Benevenuto F, Weber I (2016) Analyzing the targets of hate in online social media. In: Tenth International AAAI conference on web and social media.
49. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 15(1):1929–1958.
50. Sun C, Shrivastava A, Singh S, Gupta A (2017) Revisiting unreasonable effectiveness of data in deep learning era. In: Proceedings of the IEEE international conference on computer vision, pp 843–852.
51. Sundermeyer M, Schlüter R, Ney H (2012) LSTM neural networks for language modeling. In: Thirteenth annual conference of the international speech communication association.
52. Sutskever I, Vinyals O, Le QV (2014) Sequence to sequence learning with neural networks. In: Advances in neural information processing systems, pp 3104–3112.
53. Suzor N, Van Geelen T, Myers West S (2018) Evaluating the legitimacy of platform governance: a review of research and a shared research agenda. Int Commun Gaz 80(4):385–400.
54. The Online Hate Index: Innovation Brief (2018) Technical report, the anti-defamation League’s center for technology and society. https://www.adl.org/media/10894/download.
55. Turian J, Ratino L, Bengio Y (2010), Word representations: a simple and general method for semi-supervised learning. In: Proceedings of the 48th annual meeting of the association for computational linguistics. Association for Computational Linguistics, pp 384–394.
56. Wang B, Wang A, Chen F, Wang Y, Kuo C-CJ (2019) Evaluating word embedding models: methods and experimental results. APSIPA Trans Signal Inf Process 8:e19.
57. Wang D, Nyberg E (2015) A long short-term memory model for answer sentence selection in question answering. In: Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing (volume 2: short papers), vol 2, pp 707–712.
58. Wang W, Chen L, Thirunarayan K, Sheth AP (2014) Cursing in english on twitter. In: Proceedings of the 17th ACM conference on computer supported cooperative work & social computing. ACM, pp 415–425.
59. Weinberger KQ, Saul LK (2009) Distance metric learning for large margin nearest neighbor classification. J Mach Learn Res 10(Feb):207–244.
60. Xiang G, Fan B, Wang L, Hong J, Rose C (2012) Detecting offensive tweets via topical feature discovery over a large scale twitter corpus. In: Proceedings of the 21st ACM international conference on Information and knowledge management. ACM, pp 1980–1984.
61. Zeiler MD, Fergus R (2014) Visualizing and understanding convolutional networks. In: European conference on computer vision. Springer, pp 818–833.
62. Zhang KW, Bowman SR (2018) Language modeling teaches you more syntax than translation does: lessons learned through auxiliary task analysis. arXiv preprint arXiv:1809.10040.
63. Zhang Z, Luo L (2019) Hate speech detection: a solved problem? The challenging case of long tail on twitter. Semant Web 10(5):925–945.

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.
Md Abul Bashar received his Ph.D. degree from Queensland University of Technology (QUT), Brisbane, Australia, in 2017 and B.Sc. (Engg.) degree from Shah Jalal University of Science and Technology (SUST), Sylhet, Bangladesh, in 2008. He is an expert in Machine Learning, Deep Learning, Artificial Intelligence, and Data Mining. He works as a Postdoctoral Research Associate at QUT. He has more than seven years research and development experience at QUT data science discipline. This discipline conducts impactful theoretical research and real-life applicative projects. He worked in several projects funded by the Australian Research Council and industry partners. These projects had a practical impact in the data science community and industry. Some outcomes of these projects have been published in high-quality journals and conferences, and others have benefited our industry partners.

Richi Nayak is currently an Associate Professor (Reader) in the School of Computer Science at Queensland University of Technology (QUT). She is the Applied Data Science Program Leader of the University Centre for Data Science (CDS). She is an internationally recognised expert in data mining, text mining and web intelligence. She has combined knowledge in these areas very successfully with diverse disciplines such as Social Science, Science, and Engineering for technology transfer to real-world problems to change their practices and methodologies. Her particular research interests are machine learning and in recent years she has concentrated her work on text mining, personalization, automation, and social network analysis. She has published high-quality conference and journal articles and highly cited in her research field. She has received a number of awards and nominations for teaching, research and service activities.

Nicolas Suzor researches the regulation of networked society. He is a Professor the Law School at Queensland University of Technology, and a Chief Investigator of QUT’s Digital Media Research Centre. He is also a Chief Investigator of the ARC Centre of Excellence for Automated Decision-Making and Society. His research examines the governance of the internet and social networks, the regulation of automated systems, digital copyright, and knowledge commons. He is the author of Lawless: the secret rules that govern our digital lives (Cambridge, 2019). Nic teaches intellectual property and technology law at QUT. He is an award winning educator, receiving QUT’s David Gardiner Teacher of the Year medal in 2016 and was nationally recognised as a recipient of an Australian Awards for University Teaching Citation for Outstanding Contributions to Student Learning in 2017 for his engaging and innovative teaching.