Class-based Prediction Errors to Detect Hate Speech with Out-of-vocabulary Words

Joan Serrà
Telefónica Research,
Barcelona, Spain

Ilias Leontiadis
Telefónica Research,
Barcelona, Spain

Dimitris Spathis
Aristotle University,
Thessaloniki, Greece

Gianluca Stringhini
University College London,
United Kingdom

Jeremy Blackburn
Telefónica Research,
Barcelona, Spain

Athena Vakali
Aristotle University,
Thessaloniki, Greece

Abstract

Common approaches to text categorization essentially rely either on n-gram counts or on word embeddings. This presents important difficulties in highly dynamic or quickly-interacting environments, where the appearance of new words and/or varied misspellings is the norm. A paradigmatic example of this situation is abusive online behavior, with social networks and media platforms struggling to effectively combat uncommon or non-blacklisted hate words. To better deal with these issues in those fast-paced environments, we propose using the error signal of class-based language models as input to text classification algorithms. In particular, we train a next-character prediction model for any given class, and then exploit the error of such class-based models to inform a neural network classifier. This way, we shift from the ability to describe seen documents to the ability to predict unseen content. Preliminary studies using out-of-vocabulary splits from abusive tweet data show promising results, outperforming competitive text categorization strategies by 4–11%.

1 Introduction

The first steps in automatic text categorization rely on the description of the document content. Typically, such content is characterized by some sort of word, stem, and/or n-gram counts (e.g. Wang and Manning, 2012) or, more recently, by unsupervised or semi-supervised word/sentence/paragraph embeddings (e.g. Arora et al., 2017). While the common pipeline performs well for a myriad of tasks, there are a number of situations where a large ratio of seen vs. unseen tokens threatens the performance of the classifier. This is the case, for instance, with abusive language in online user content or microblogging sites (Nobata et al., 2016; Mehdad and Tetreault, 2016). In such cases, the volume of annotated data is not massive, and the vocabulary changes rapidly and non-consistently across sites and user communities. Sometimes these changes and inconsistencies are intentional, so as to hide the real meaning from traditional automatic detectors using hate-word dictionaries or blacklists. For example, the notorious online community *4chan* launched "Operation Google", which aimed to replace racial slurs on social media with the names of prominent tech companies in a sort of adversarial attack on Google's Jigsaw project (Hine et al., 2017).

To avoid relying on specific tokens, existing text classification approaches can incorporate so-called linguistic features (Brody and Diakopoulos, 2011), such as number of tokens, length of tokens, number of punctuations, and so on, or syntactic features (Nobata et al., 2016), based on part-of-speech tags, dependency relations, and sentence structure. Distributional representations (Le and Mikolov, 2014) and recurrent neural network (RNN) language models (Mikolov, 2012) are also used, together with more classical approaches involving word or character n-grams (Wang and Manning, 2012).

The task of automatic hate speech detection is usually framed as a supervised learning problem. Surface-level features like bag-of-words and embeddings systematically produce reasonable clas-

sification results (Chen et al., 2012; Nobata et al., 2016). Character-level approaches perform better than token-level ones, since the rare spelling variations of slang and swear words will result in unknown tokens (Mehdad and Tetreault, 2016). More complex features like dependency-parse information or specific linguistic attributes like politeness-imperatives have been effective (Chen et al., 2012). Also lexical resources like lists of slurs, are proven effective but only in combination with other features (Schmidt and Wiegand, 2017).

2 Proposed approach

We propose to perform text categorization following a two step approach (Fig. 1). Firstly, a language model for next-character prediction is trained with the data corresponding to each single category. Secondly, we use a normalized, sequential measurement of the performance of those language models as input to a neural network classifier. In a sense, we aim to shift from 'what is said' (ability to describe) to 'the way of saying' (ability to predict). The idea is that the language model should be tailored to a given category, but without developing a strong dependence on particular characters that are frequent or representative for that category.

The concept of class-based errors is reminiscent of universal background models in speaker verification (Reynolds et al., 2000) and a few document attribution strategies in information retrieval (Serrà et al., 2012). Using error signals for classification also relates to derivative-based similarity and classification in time series (Keogh and Pazzani, 2001).

2.1 Class-conditioned language model

For the class-conditioned language model we use a character-based RNN. Sequences of one-hot encoded characters $\mathbf{X}$ are passed through a time-distributed dense layer with parametric rectified linear unit (PReLU) activation (He et al., 2015), which form a preliminary embedding. This is shared with the subsequent layers. The intuition behind using a PReLU activation after the embedding stems from the fact that it can only improve the results or, at worst, be automatically bypassed: if a linear transformation (no activation) is really the best option, the model can still learn it (He et al., 2015).

For each class i, one gated recurrent unit (GRU; Cho et al., 2014) is used, followed by a time-distributed dense layer with softmax output, yielding the next-character prediction sequence $\hat{\mathbf{X}}^{(i)}$. With that and the delayed version of $\mathbf{X}$, we calculate a class-conditioned error sequence $\mathbf{e}^{(i)}$ for each character (and zero-mask it according to the sequence length, if needed). In our experiments with one-hot encoded character inputs, $\mathbf{e}^{(i)}$ corresponds to categorical cross-entropy. The final loss is taken to be

$$\mathcal{L}\left(y, \mathbf{e}^{(1)}, \dots \mathbf{e}^{(c)}\right) = \sum_{i=1}^{c} \left(\frac{\mathbb{1}_{\{i\}}(y)}{n-1} \sum_{j=1}^{n-1} e_j^{(i)} \right),$$

where y is the class label of the instance, c is the total number of classes, n is the sequence length of the instance, and $\mathbb{1}()$ is an indicator function.

2.2 Normalization

To perform classification, we take the class-conditioned error sequences $\mathbf{e}^{(i)}$ provided by the language model, and concatenate them into a matrix $\mathbf{E} = [\mathbf{e}^{(1)}, \dots \mathbf{e}^{(c)}]$ for each instance. We then use an instance normalization layer

$$\bar{\mathbf{E}} = \sigma\left(g\mathbf{E}/\mu\right),$$

where $\sigma()$ is an activation function, g is a gain constant, and μ is the average over all the elements of $\mathbf{E}$ (taking sequence masking into account). In principle, we could learn g (or extended vector/matrix versions of it) and use any activation. However, we empirically found $g = 1/3$ and a sigmoid activation to work well enough with character-based cross-entropy sequences.

2.3 Classification

After computing $\bar{\mathbf{E}}$, we input it to a two-layer neural network with PReLU activations, dropout (Srivastava et al., 2014), and a softmax output. This architecture decision is motivated by the fact that we need at least one layer of an another neural network to form the binary classification, since after the language model we have as many errors as characters. A second layer is added to allow the model to perform nonlinear classification based on the error sequences. Regarding normalization, our initial experiments involved no normalization and the performance was much poorer, to the level of the considered alternatives or slightly below.

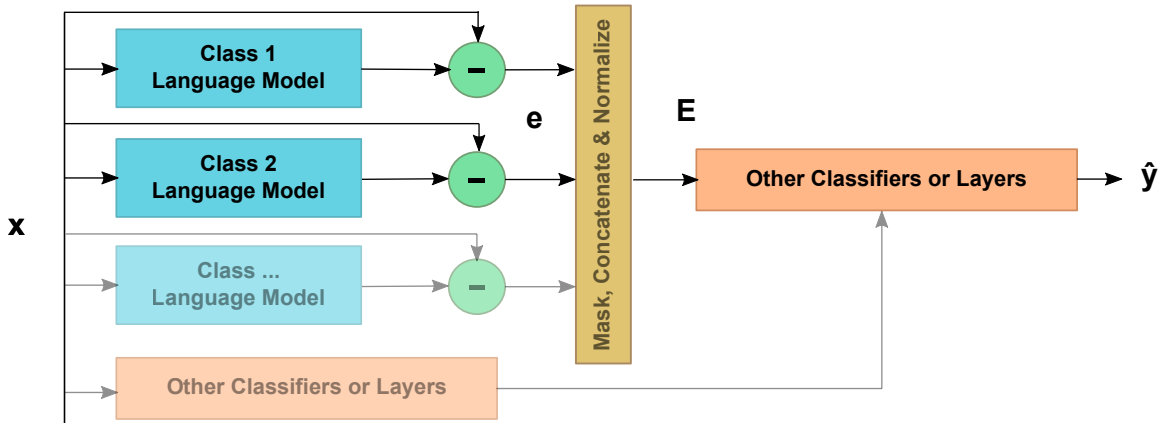

Figure 1: General schema of the proposed model. First, a language model for next-character prediction is trained with the data corresponding to each single class. Second, a normalized, sequential measurement of the performance of those language models is fed as input to a neural network classifier.

2.4 Model setup

We train our models using Adam gradient descent (Kingma and Ba, 2015) until we do not see an improvement on the validation set for 4 epochs. For the language model, we use layer dimensionalities 250 (embedding) and 500 (GRU). For the classification model, we use a dropout of 0.5 and dimensionality 100. At classification time, the language model's weights remain frozen. To implement our models we used Keras (Chollet, 2015) with Theano (Theano Development Team, 2016).

3 Evaluation data

To evaluate the suitability of the proposed concepts, we choose the task of detecting abusive tweets with out-of-vocabulary (OOV) words, using a sample of 2 million tweets from the Twitter 1% feed. Half of the sample is selected based on their use of 'hate words' from a crowdsourced dictionary[1], while the other half is selected randomly. We manually filter the dictionary to remove overly common and ambiguous words like "india", a localized slur for dark skinned people.

Due to lack of baselines and datasets that take into account OOV words, we create semi-synthetic splits with OOV words, in order to simulate data with new or heavily misspelled tokens. We randomly perform 10 train/validation/test splits, compute the area under the receiver operating characteristic curve (AUC), and report the mean and standard deviation over the 10 test splits. Two setups are considered:

- Easy – For the positive class, we take the hate-word list and split it into 70% for training and 15% for validation and testing. In performing such split, we force words with the same stem to end up in the same split. We then select tweets from the entire corpus that contain at least one stemmed word from each split. For the negative class, we just select randomly from the remaining tweets until we have balanced train/validation/test sets.

- Hard – Besides the list of hate words, we also consider a list of common words (top 1,000 to 3,000 most frequent English words[2]). We proceed as with the positive class of the Easy setup, and generate balanced train/validation/test splits for both abusive and non-abusive tweets. In addition, to increase difficulty, we remove tweets with list words appearing in more than one split (that is, we ensure that the intersection of listed words is null between train/validation/test).

To the best of our knowledge, there are no established benchmark datasets for the problem of OOV word detection. Recent work on characterizing OOV words in twitter attempted to build a dataset of such words (Maity et al., 2016), but mostly focused on content analysis and categorization (e.g., *wassup* and *iknow*, belong to word mergings). We plan to develop big crowd-sourced datasets of OOV social media texts and provide them free to the research community.

[1] http://hatebase.org

[2] http://www.ef.com/english-resources/english-vocabulary/

Setup	Word NB	Char NB	Word NB-LR	Char NB-LR	Char RNN	Proposed
Easy	0.900 (0.091)	0.849 (0.052)	0.912 (0.113)	0.902 (0.102)	0.858 (0.141)	**0.951** (0.080)
Hard	0.634 (0.109)	0.663 (0.080)	0.580 (0.089)	0.679 (0.038)	0.619 (0.101)	**0.705** (0.059)

Table 1: AUC scores for the considered baselines (see text) and the proposed approach. A null randomized model yields 0.514 (0.082) and 0.503 (0.090) for the Easy and Hard setups, respectively.

4 Results

We compare the proposed approach with 5 of the most common and competitive baselines in abusive language detection (Djuric et al., 2015; Mehdad and Tetreault, 2016). The first two are based on a naïve Bayes classifier using the TF-IDF of both word and character n-grams (NB). The next two are based on the approach proposed by Wang and Manning (2012): similarly, word and character n-grams are constructed, but NB ratios are calculated, and logistic regression is used as a classifier (NB-LR). The rationale behind this choice over other classifiers (e.g SVMs) was that Wang and Manning (2012) found that for short text sentiment tasks, NB actually does performs better than SVMs. For the previous non-neural count-based baselines, we use all combinations of {1,2,3}-grams with cutoff frequencies of 20 (NB) and 100 (NB-LR). These cutoff frequencies were chosen in-sample in order to maximize the performance of these baselines. The fifth baseline is a character-based RNN using a time-distributed embedding layer, followed by a PReLU activation, a GRU, a dense layer with PReLU activation, and a dense layer with sigmoid output. We try different values for the dimensionality of the aforementioned layers and finally use 200 (embedding), 400 (GRU), and 200 (dense).

The result of the comparison is reported in Table 4. We see that, among the 5 baselines, there is no clear winner for the two setups. Word-based NB-LR performs best in the Easy setup and character-based NB-LR performs best in the Hard setup. Nonetheless, the proposed approach outperforms them by 4.2 and 3.8%, respectively. Compared to the average baseline performance, we observe a relative improvement of 7.5 and 11.0%. We also note that the standard deviation of the proposed approach (across runs) is comparatively low with respect to the baselines.

5 Future work

In this paper, we deal with hate speech detection and, in particular, with abusive OOV words. To this extent we propose to use the error signal of class-based language models as input to text classification algorithms. In particular, we train a next-character prediction model for any given class, and then exploit the error of such class-based models to inform a neural network classifier. This way, we intend to shift from the 'ability to describe seen documents to the 'ability to predict unseen content. Experiments using OOV splits from abusive tweet data show promising results, outperforming competitive text categorization strategies by 4–11%.

We envision a number of potential extensions for the proposed approach: adding an 'all-class' predictor to the language models, improve (or learn) the error sequence normalization, studying the effect of adding further classifiers in parallel to the proposed classification model, ways of fusing those, play with class-based sentence or paragraph embeddings, etc. Generalizing the architecture to longer sequences is a main task for further research, perhaps considering recurrent, quasi-recurrent (Bradbury et al., 2017), or convolutional networks for the classification stage. A qualitative analysis of the output of above classifiers is planned as future work. Due to the fact that our data are weakly-labeled and come from dictionaries, we also plan to shift our focus to real-world, curated data sets of abusive language, as well as evaluate our models on human-annotated crowd-sourced data.

Acknowledgments

This work has been fully funded by the European Commission as part of the ENCASE project (H2020-MSCA-RISE of the European Union under GA number 691025).

References

S. Arora, Y. Liang, and T. Ma. 2017. A simple but tough to beat baseline for sentence embeddings. In *Proc. of the Int. Conf. on Learning Representations (ICLR)*. https://openreview.net/pdf?id=SyK00v5xx.

J. Bradbury, S. Merity, C. Xiong, and R. Socher. 2017. Quasi-recurrent neural networks. In *Proc. of the Int. Conf. on Learning Representations (ICLR)*. http://arxiv.org/abs/1611.01576.

S. Brody and N. Diakopoulos. 2011. Cooooooooooooooollllllllllllll!!!!!!!!!!!!!! Using word lengthening to detect sentiment in microblogs. In *Proc. of the Conf. on Empirical Methods in Natural Language Processing (EMNLP)*. pages 562–570.

Ying Chen, Yilu Zhou, Sencun Zhu, and Heng Xu. 2012. Detecting offensive language in social media to protect adolescent online safety. In *Privacy, Security, Risk and Trust (PASSAT), 2012 International Conference on and 2012 International Confernece on Social Computing (SocialCom)*. IEEE, pages 71–80.

K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio. 2014. On the properties of neural machine translation: encoder-decoder approaches. In *Proc. of the Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST)*. pages 103–111.

Franois Chollet. 2015. keras. `https://github.com/fchollet/keras`.

N. Djuric, J. Zhou, R. Morris, M. Grbovic, V. Radosavljevic, and N. Bhamidipati. 2015. Hate speech detection with comment embeddings. In *Proc. of the Int. Conf. on World Wide Web (WWW)*. pages 29–30.

K. He, X. Zhang, S. Ren, and J. Sun. 2015. Delving deep into rectifiers: surpassing human-level performance on ImageNet classification. In *Proc. of the IEEE Int. Conf. on Computer Vision (ICCV)*. pages 1026–1034.

G Hine, J Onaolapo, E De Cristofaro, N Kourtellis, I Leontiadis, R Samaras, G Stringhini, and J Blackburn. 2017. Kek, cucks, and god emperor trump: A measurement study of 4chan's politically incorrect forum and its effects on the web. In *International Conference on Web and Social Media (ICWSM)*. Association for the Advancement of Artificial Intelligence (AAAI).

E. Keogh and M. Pazzani. 2001. Derivative dynamic time warping. In *Proc. of the SIAM Int. Conf. on Data Mining (SDM)*.

D. P. Kingma and J. L. Ba. 2015. Adam: a method for stochastic optimization. In *Proc. of the Int. Conf. on Learning Representations (ICLR)*. https://arxiv.org/abs/1412.6980.

Q. Le and T. Mikolov. 2014. Distributed representations of sentences and documents. In *Proc. of the Int. Conf. on Machine Learning (ICML)*. volume 32, pages 1188–1196.

Suman Maity, Anshit Chaudhary, Shraman Kumar, Animesh Mukherjee, Chaitanya Sarda, Abhijeet Patil, and Akash Mondal. 2016. Wassup? lol: Characterizing out-of-vocabulary words in twitter. In *Proceedings of the 19th ACM Conference on Computer Supported Cooperative Work and Social Computing Companion*. ACM, pages 341–344.

Y. Mehdad and J. Tetreault. 2016. Do characters abuse more than words? In *Proc. of the Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*. pages 299–303.

T. Mikolov. 2012. *Statistical language models based on neural networks*. Ph.D. thesis, Brno University of Technology.

C. Nobata, J. Tetreault, A. Thomas, Y. Mehdad, and Y. Chang. 2016. Abusive language detection in online user content. In *Proc. of the Int. Conf. on World Wide Web (WWW)*. pages 145–153.

D. A. Reynolds, T. F. Quatieri, and R. B. Dunn. 2000. Speaker verification using adapted Gaussian mixture models. *Digital Signal Processing* 10(1-3):19–41.

Anna Schmidt and Michael Wiegand. 2017. A survey on hate speech detection using natural language processing. *SocialNLP 2017* page 1.

J. Serrà, H. Kantz, X. Serra, and R. G. Andrzejak. 2012. Predictability of music descriptor time series and its application to cover song detection. *IEEE Trans. on Audio, Speech and Language Processing* 20(2):514–525.

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15:1929–1958.

Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints* abs/1605.02688. http://arxiv.org/abs/1605.02688.

S. Wang and C. D. Manning. 2012. Baselines and bigrams: simple, good sentiment and topic classification. In *Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL)*. volume 2, pages 90–94.