CSR: Character-level Sequence Representation for Classification of Tweets with Misspellings

CIL Project 2 Report - Group: CGXS

Xiaochen Zheng
ETH Zürich
xzheng@student.ethz.ch

Guillaume Wang
ETH Zürich
guiwang@student.ethz.ch

Costanza Maria Improta
ETH Zürich
cimprota@student.ethz.ch

Stevan Mihajlovic
ETH Zürich
mstevan@student.ethz.ch

***Abstract*—With social media being at the centre of instant communication in today's world, people turn to Twitter to convey their thoughts and opinions concisely. Text sentiment analysis aims at accurately depicting them.**

The characteristics of tweets complicate the use of word-level methods, as user-specific variations in spelling and abbreviations preclude the direct encoding of words to indices. To circumvent this requires significant effort at the preprocessing stage.

In this work, we propose CSR, a novel language model based on character-level embedding, which overcomes the aforementioned challenges of user-generated text, while being competitive with state-of-the-art methods in correctly classifying sentiment into positive and negative class.

Our study compares CSR with more commonly used word-level methods, and it demonstrates the suitability of character-level embedding to the task. Additionally, we propose a combination of the two embedding methods, achieving our best performance accuracy due to the complementary nature of their outputs.

I. Introduction

Online social media are at the centre of social life in today's world. Platforms such as Twitter are used to express points of views on topics as diverse as politics, world news, celebrities or television shows. With 500 millions of new tweets published each day[1], tweets are an ideal playground for Natural Language Processing (NLP) on big data. At the same time, their diversity, informativeness and wide audience give a high practical importance to the challenge of efficiently and accurately processing the tweets.

Over the last ten year, several methods have been proposed to tackle the problem of sentiment analysis, i.e to classify texts as expressing a positive or a negative sentiment. A notable instance is the IMDb movie review classification task [1], commonly used in the field as a performance benchmark.

However, for datasets of tweets, the users' frequent unconventional use of spelling and abbreviations, due to the 140-character limit or just carelessness, heavily complicates information extraction, and consequently also the sentiment analysis.

[1]Twitter web-page: https://business.twitter.com/

In this work, we propose a novel language model based on character-level embedding, which overcomes the aforementioned challenges of user-generated text, while being competitive with state-of-the-art methods.

II. Related Work

Recent work on text sentiment analysis focuses on leveraging methods for word and sequence representation, i.e embedding a word or a sentence into a vector in a meaningful way. Then any vectorial classifier can be used for the sentiment classification task. The most frequently used methods are *word-level* approaches, so that the tokens (the units of the sequence to embed) are taken to be the words.

By contrast, our proposal is inspired by the work of Zhang *et al.* on *character-level* convolutional neural networks [2]. The main innovations we introduce are the use of residual blocks, of a recurrent neural network (RNN); as well as the evaluation on a dataset of short texts with misspellings and abbreviations (tweets), instead of longer texts (reviews, articles, or news). Details are in Section III.

For comparison with word-level based methods, we also implement and evaluate the dataset on classifiers using two classic word-level models, respectively GloVe and BERT.

GloVe (**Glo**bal **Ve**ctors) is an unsupervised learning algorithm for obtaining vector representations for words [3], which combines global matrix factorization and local context window methods. The algorithm infers the word embeddings from statistics of word cooccurrences in a corpus.

BERT (**B**idirectional **E**ncoder **R**epresentations from **T**ransformers) is a deeply bidirectional, unsupervised language representation which takes into account both the occurence and the position of a given word in the context. The model is pretrained for the Mask LM task, which randomly replaces some input tokens (words) with [MASK], and asks to predict what the original tokens were.

Similarly to our final experiment, Liang *et al.* [4] also combines word-level and character-level approaches, finding that aggregating the two allows for great improvement.

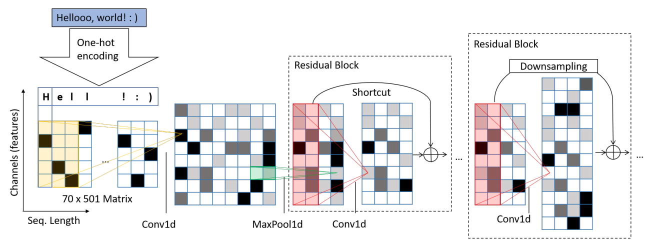

Figure 1. Architecture of Character-level Sequence Representation (CSR).

III. Proposed Method

A. *Character-level Sequence Representation (CSR) with Multi-layer Bidirectional RNN*

Zhang *et al.* [2] stated that character-level CNNs work well for user-generated data without the need for word embeddings. Instead of requiring long corpuses to populate large vocabularies (i.e. BERT uses a 30'000-word vocabulary [5]), character-level embedding only requires an alphabet for one-hot encoding of characters. The alphabet used in our work is the same as the one proposed by Zhang *et al.* [2] and consists of 70 characters, divided into 26 English letters, 10 digits, 33 other characters and the new line character:

```
abcdefghijklmnopqrstuvwxyz0123456789
 -,;.!?:'''/\|_@#$%^&*~`+-=<>()[]{}
```

Here we propose the Character-level Sequence Representation (CSR), a novel character-level embedding model, which is inspired by Zhang's Character-level Convolutional Networks (CharCNN) [2]. The main components of the model, shown in Fig.1, are one-dimensional convolution (Conv1d) and batch normalization (BN). Each BN is implemented right after each Conv1d, and it is then followed by a ReLU activation layer.

Layer	Input Channels	Output Channels	Module
1	70	256	Bottleneck
2	256	256	Residual
3	256	256	Residual
4	256	512	Residual
5	512	512	Residual
6	512	512	Maxpool

Table I
STRUCTURE OF CSR-RES-4-D.

1) Bottleneck Module: The bottleneck module consists of one Conv1d with a kernel of size 7 and stride equal to 1. After the convolutional layer, a maxpooling of size 3 is applied.

2) Residual Module: The residual module includes a Conv1d (residual) layer with kernel of size 7 and an *identity shortcut*. If the size of the feature maps (along the sequence length axis in Fig.1) is halved, the output channels (number of kernels) of the Conv1d need to be doubled in order to correctly match the dimensions. This is achieved by Conv1d with kernel size of 1 (downsampling in Fig.1) for the shortcut.

The convolutional layers have stride 1 and pooling filters are all non-overlapping. The design of our CSR-Res-4-d model is given in Tab. I, where "*Res*" stands for residual module, *4* is the number of residual modules and "*d*" stands for downsampling.

B. *Word-level Sequence Representation (WSR) with Multi-layer Bidirectional RNN*

As both an add-on to our proposed method and as a mean of comparison, we implemented some classifiers using word-level sequence representation methods.

For instance, GloVe may be used as a word embedding algorithm for sequence representation by simply concatenating the vectors of the words in the sequence; additional processing by convolutional layers or RNN are then used to obtain a fixed-length representation. For our experiments, we use pre-trained 200-dimensional word-vectors obtained by GloVe on a dataset of tweets without fine-tuning it to our dataset, and a succession of convolutional layers and RNN as shown in Fig.2.

Accordingly, we use the same data-preprocessing method as for pretraining the GloVe embedding (publicly available as Ruby script on the GloVe web-page). [2] This is followed by a word segmentation step. This preprocessing step helps minimizing out-of-vocabulary occurrences.

C. *Combination of Character- and Word-level Embedding (CWSR)*

In an attempt to aggregate the two models, we apply a Conv1d layer to the output from both the CSR and the word-level embedding, and then concatenate them into a 600-dimensional vector representation. We aim to combine the two models and allocate equal weights for both models. With this representation, consisting of character-level embedding and word-level embedding, a fully-connected dense layer is used to do the classification task. The overall structure is shown in Fig.2.

The Multi-layer Bidirectional RNN we use in all experiments are either Gated Recurrent Unit (GRU) [6] or Long Short-Term Memory (LSTM) [7].

IV. Baselines

The baselines implemented with word embedding are BoW, Mean- and SIF-embedding and BERT (with a GRU classifier). On the other hand, CharCNN is developed as baseline with character level embedding.

[2]GloVe web-page: https://nlp.stanford.edu/projects/glove/

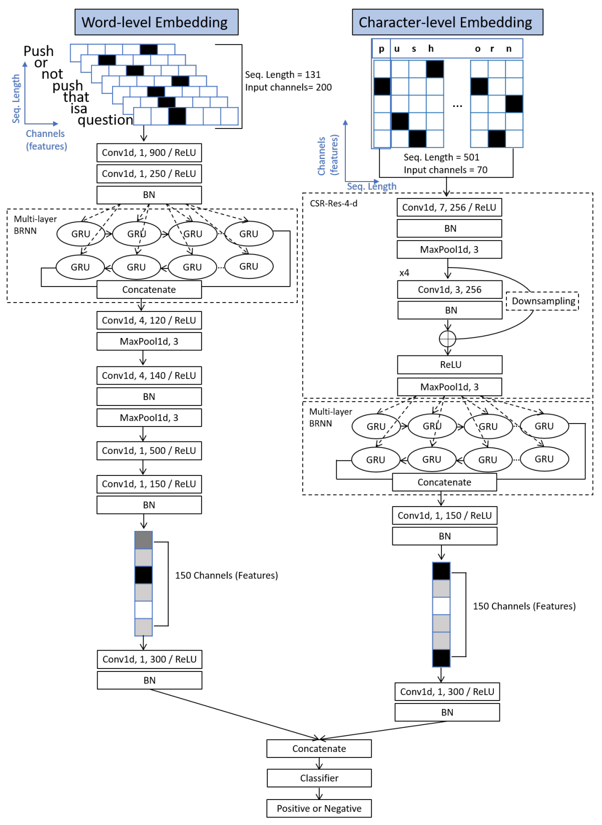

Figure 2. Architecture of Character- and Word-level Sequence Representation (CWSR). The convolutional layer is shown in the format: (Conv1d, kernel size, output channels). The strides of Conv1d and MaxPool1d are 1 and 3 respectively. The final classifier is a two-layer fully-connected neural network with 1000 hidden neurons with ReLU activation, followed by a sigmoid.

A. *Bag of Words (BoW)*

Our first approach focuses on exploring different vocabularies for our text corpus using FFNN with ReLu activation functions. We explore the significance of the vocabulary by using the Bag-of-Words (BoW) method and applying it to our own predefined vocabularies.

We first define a vocabulary V as an ordered set of words. We represent tweet as a vector of length $|V|$ - one dimension for each word of the vocabulary. Every word in the vocabulary is one-hot encoded. The resulting tweet vector t is a sum of the individual word vectors.

The bottleneck of this model is the size of the vocabulary as it negatively impacts the performance and increases runtime complexity. Using different vocabularies does not capture semantic relation between words.

B. *Mean-embedding*

The second baseline makes use of algorithms for classification of vectorial data, an extensively studied problem in the machine-learning literature. However, those algorithms require fixed-length vectors as inputs, while varying-length representations of tweets are easily obtained by concatenating the word embeddings. A simple way to circumvent this is to take the mean of the word embeddings in the tweet. Thereafter, a vectorial classifier can be used to get the polarity class.

For this baseline, we use the same pretrained GloVe word embeddings and the same preprocessing as in Section III-B, as well as a vanilla linear classifier, namely scikit-learn's `SGDClassifier` with default parameters [8].

C. *SIF-embedding*

The previous baseline may be refined by taking a weighted average of the word vectors, rather than the unweighted average, to represent each tweet. Thus, the third baseline makes use of SIF (smooth-inverse-frequency), a simple yet promising weighting scheme proposed by Arora *et al.* [9].

V. Experiments

For the benchmarking, we first trained the three baseline models introduced above (BoW, Mean- and SIF-embedding), pretrained BERT with GRU on preprocessed dataset (BERT-GRU), and Character-level embedding with some dense layers as a classifier (CharCNN). We then develop our novel model by modifying the CharCNN implementation, replacing the Conv1d layers and MaxPool layers with residual layers and downsampling layers as shown in Fig.1. CSR-Res-4 denotes the model using the same classifier as CharCNN. We then experimented with different classifiers (GRU and LSTM) and different numbers of residual modules (4 and 6) to improve accuracy, as shown in table II. All character-level models were trained on raw data, while word-level models were trained on preprocessed data as described in Sections III and IV. All models were trained without any fine-tuning strategy, such as learning rate decay. The full dataset were randomly spiltted into training (90%) and validation (10%) set.

The accuracy reported in the table is based on the public test set made available on the Tweet Sentiment Classification task's corresponding Kaggle competition. [3]

VI. Discussion

For datasets of tweets, frequent use of abbreviations and misspellings heavily complicates information extraction. Indeed, as we saw, many NLP algorithms start by encoding words as their indexes based on a pre-chosen ordered vocabulary. When the text contains misspelled or abbreviated words, or even domain-specific tokens (e.g *"rt"* for *"re-tweet"*, *"#"* for *"hashtag"*), then some preprocessing is needed in order to map them back to their original spelling. This is exemplified by the poor performance of the BoW baseline without preprocessing.

Both mean-embedding and SIF-embedding perform dramatically worse than the more complex models, but better

[3]https://www.kaggle.com/c/cil-text-classification-2020/data

Model	Preprocessing	Accuracy	Model Size
BoW	None	**76.06%**	5.0M
Mean-embedding	GloVe-twitter	**78.62%**	-
SIF	GloVe-twitter	**77.52%**	-
BERT-GRU	None	**84.68%**	3.5M
BERT-GRU	Misspellings Replaced	**86.32%**	3.5M
CharCNN	None	**84.90%**	6.6M
CSR-Res-4	None	85.90%	6.0M
CSR-Res-4-GRU	None	85.86%	3.5M
CSR-Res-4-d-GRU	None	86.22%	3.5M
CSR-Res-4-d-LSTM	None	86.34%	3.5M
CSR-Res-6-GRU	None	86.52%	3.9M
CSR-Res-6-d-GRU	None	**86.86%**	3.9M
WSR	GloVe-twitter	**87.78%**	1.9M
CWSR	None/GloVe-twitter	**87.88%**	3.9M

Table II
EXPERIMENTAL RESULTS

than BoW. This was expected, since they employ the same reasoning behind BoW but with a more articulated encoding.

On the other hand, surprisingly, SIF-embedding perform worse than the unweighted variant on this specific task. We hypothesize that this is because less weight is put on common negative words, so that, for example, the tweet *"This is extremely not good"* is less distinguishable from *"This is extremely good"*. This effect of SIF on sentiment analysis was not reported by the authors of the method, despite using the same kind of vectorial classifier. The difference may be due to the shortness of the tweets, compared to the benchmarks used by Arora *et al.* [9], making the method more sensitive to this effect.

BERT is the current state-of-the-art model concerning word-level embeddings. Along with a multi-layer BRNN, our implementation trained on raw data scored 84.68%, thus outperforming the other methods in the same category. Interestingly, the same implementation, trained on preprocessed data with mispellings replaced, yielded a higher accuracy. This shows that BERT's performance is indeed negatively affected by use of abbreviations and misspellings.

These considerations are the main motivation behind a character-level approach to tweet sentiment analysis. Furthermore, our implementation of CharCNN outperformed BERT, with both being trained on raw data. This may be see as a hint that character-level embedding might be more suitable to the task of classifying user-generated text than word-level embedding.

The aforementioned concepts lead us to propose the CSR model. The boost in performance CSR has, compared to CharCNN, is given by the ResNet layers. In fact, our results show that they help with retaining more information from earlier layers while preventing an earlier overfitting of the model. These claims find further proof when we increase the number of residual layers from 4 to 6, since the performance consequently increases. Concerning the classification method to append to CSR, we tried both LSTM and GRU. As they both yelled similar results, we opted to refine our implementation using a GRU classifier as it is proven to be computationally cheaper [10].

Our combined model introduced in Section III-C, CWSR, combines both our novel CSR model and a word-level model (WSR). The motivation behind this is that the two parts yield complementary features, and the concatenation of the two outputs contains more information than either one of them. Indeed, our final experiments show some improvement. However, in contrast with Liang *et al.* [4], the improvement we observe is small. Further experiments are needed to understand the difference between our setting and theirs.

VII. CONCLUSION

In this work we presented a purely character-level sequence representation model that is particularly useful for the specific task of user-generated sentiment classification. Experiments showed that our CSR model performed as well on raw data as word-level-based models on preprocessed data. Indeed, character-level embedding avoids the issue of out-of-vocabulary words caused by misspellings or abbreviation in word embedding models.

Directions for future work

As our final experiment shows, combining our CSR model with a standard word-level-based model does not lead to significant improvement. Further experiments are needed to understand why, and to decide between hypotheses:

1) The CSR yields features that are similar to what the word-level approach gives, so that combining the two groups of features does not help.
2) The combined network (CWSR) gives too much importance to one of its two branches, so that the information flow from one is drowned by the other. Then there may be problems of vanishing gradients during training.

In the latter case, a modification of the aggregation method (the final layer of the CWSR) should suffice to observe changes in performance. A simple way to ensure equal information flow from both branches would be to batch-normalize the outputs of each branch and take their sum instead of their concatenation. An alternative way, used by Liang *et al.* [4], would be to use a highway network for combination.

Finally, other ways to combine character- and word-based sequence representations can be explored. For example, sequence representations can be concatenated slot-wise while preserving temporal order, i.e before flattening or taking the hidden states of a RNN. This could lead to a sequence representation method that both leverages word embeddings, and correctly incorporates punctuation, abbreviations and out-of-vocabulary words.

References

[1] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, June 2011, pp. 142–150.

[2] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Advances in Neural Information Processing Systems 28 (NIPS 2015)*, 2015, pp. 649–657.

[3] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543.

[4] D. Liang, W. Xu, and Y. Zhao, "Combining word-level and character-level representations for relation classification of informal text," in *Proceedings of the 2nd Workshop on Representation Learning for NLP*, 2017, pp. 43–47.

[5] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018.

[6] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: continual prediction with lstm," in *1999 Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470)*, vol. 2, 1999, pp. 850–855 vol.2.

[7] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, p. 1735–1780, Nov. 1997.

[8] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[9] S. Arora, Y. Liang, and T. Ma, "A simple but tough-to-beat baseline for sentence embeddings," in *5th International Conference on Learning Representations (ICLR)*, 2017.

[10] W. Wang, N. Yang, F. Wei, B. Chang, and M. Zhou, "Gated self-matching networks for reading comprehension and question answering," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics, Jul. 2017, pp. 189–198.