What Did You Say? On Classification of Noisy Texts

No Author Given

No Institute Given

Abstract. Text classification is a fundamental task in natural language processing and huge body of research has been devoted to it. But one important aspect of noise robustness received relatively low attention. In this work we are bridging this gap, introducing results on noise robustness testing of modern classification architectures on English and Russian.

1 Introduction

A lot of text classification applications like sentiment analysis or intent recognition are connected to the user-generated data, where no correct spelling or grammar may be guaranteed.

Classical text vectorization approach such as bag of words with one-hot or TF-IDF encoding encounters out-of-vocabulary problem given vast variety of spelling errors. Although successful applications to low-noise tasks on common datasets [Joulin et al., 2016,Howard and Ruder, 2018], not all models behave well with real-world data like comments or tweets.

This work is organized as follows. In section 2 we describe previous works on this topic and basic units, which we used in our models. In section 3 the models in question are described. And sections 4 and 5 are describing experimental setup and obtained results with some interpretation.

2 Related Work

In recent years there were proposed some approaches to mitigate described issue. Mostly they are neural network based. These are FastText model [Joulin et al., 2016] where word embeddings for unknown word are generated on-the-fly from embeddings of constituent symbol n-grams, Robust Vectors [Malykh, 2018] where word embedding are generated basing on bag-of-letters representation or char-level models like [Kim et al., 2016] where word embedding is based on symbol embedding.

But there were a little research body on noisy texts classification with emphasis for the noise robustness. By this work we are aiming to bridge the gap and seed additional research in this area.

2.1 Gated Recurrent Unit

The Gated Recurrent Unit (GRU) is a common RNN architecture which memorizes state between timesteps thus mitigates vanishing gradients problem. It was introduced in [Cho et al., 2014].

Formally it described in Equations 1. Denoting by $\boldsymbol{x}_t$ the input vector at time t; by $\boldsymbol{h}_t$, the hidden state vector at time t; by $W_{x\cdot}$ (with different second subscripts), matrices of weights applied to the input; by $W_{h\cdot}$, matrices of weights in recurrent connections, and by $\boldsymbol{b}$ the bias vectors, we get the following formal definition:

$$\begin{aligned}
\boldsymbol{u}_t &= \sigma(W_{xu}\boldsymbol{x}_t + W_{hu}\boldsymbol{h}_{t-1} + \boldsymbol{b}_u),\\
\boldsymbol{r}_t &= \sigma(W_{xr}\boldsymbol{x}_t + W_{hr}\boldsymbol{h}_{t-1} + \boldsymbol{b}_r),\\
\boldsymbol{h}'_t &= \tanh(W_{xh'}\boldsymbol{x}_t + W_{hh'}(\boldsymbol{r}_t \odot \boldsymbol{h}_{t-1})),\\
\boldsymbol{h}_t &= (1-\boldsymbol{u}_t)\odot \boldsymbol{h}'_t + \boldsymbol{u}_t \odot \boldsymbol{h}_{t-1}.
\end{aligned} \tag{1}$$

2.2 Attention

The RNNs have commonly known flaw, they rapidly forget earlier timesteps, e.g. see [Bengio et al., 1994]. To mitigate this issue an attention mechanism was introduced. The already classic approach is described in paper [Bahdanau et al., 2014]. A soft alignment model produces weights α_{ti} that control how much each input word influences the resulting output vector. The score α indicates whether the network should be focusing on this specific word right now. υ is the text vector that summarizes all the information of words. Soft attention drastically improves translation and classification for longer sentences and is now the standard approach. More formally it is described in Equations 2.

$$\begin{aligned}
\boldsymbol{\upsilon}_t &= \tanh\left(W_\omega \boldsymbol{h}_t + \boldsymbol{b}_\omega\right)\\
\boldsymbol{\alpha}_t &= \frac{\exp\left(\boldsymbol{\upsilon}_t^T \boldsymbol{u}_i\right)}{\sum_{j=1}^{T}\exp\left(\boldsymbol{\upsilon}_j^T \boldsymbol{u}_i\right)}\\
\boldsymbol{\upsilon} &= \sum_{t=1}^{T}\boldsymbol{\alpha}_t \boldsymbol{h}_t
\end{aligned} \tag{2}$$

where W_ω and $\boldsymbol{b}_\omega$ are parameters of hidden states linear transformation; and $\boldsymbol{u}_i$ is some external vector, in case of GRU it could be hidden state h_i at the end of a sequence. $\boldsymbol{u}_i$ could be considered as a vector of context, meaning that vectors which are closer to context one should have more weight.

2.3 Character-level Convolutional Networks

Convolutional neural networks have proven to be a useful tool for text classification [Kim, 2014]. They can represent any character sequence thus solve the out-of-vocabulary problem.

The *convolution* itself is operation of Hadamard multiplication of some matrix k with patch from input A matrix of corresponding size followed by summation of the resulting matrix to one value. It is defined by the following equation:

$$b_{ij} = \sum \sum k \odot A_{[i-s_h, i+s_h]:[j-s_v:j+s_v]}$$

where b_{ij} is resulting value, k is weight matrix with size $(2 \cdot s_v + 1) \times (2 \cdot s_h + 1)$, which is called *kernel*, s_v and s_h are kernel sizes, vertical and horizontal respectively; A is an original matrix which a convolutional kernel slides over.
where b_{ij} is resulting value, k is weight matrix with size $(2 \cdot s_v + 1) \times (2 \cdot s_h + 1)$, which is called *kernel*, s_v and s_h are kernel sizes, vertical and horizontal respectively; A is an original matrix which a convolutional kernel slides over.

Working with texts for convolutions is somewhat different from pictures. The convolution of text is receiving some vector representation of a character and one size of a kernel is fixed to vector representation dimensionality. [Zhang et al., 2015]

3 Models

In our experiments we compare performance of the following models.

3.1 CharCNN

The text is represented as a sequence of one-hot symbols. This model consists of character embedding layer, convolution layer with 256 filters; kernel size is 15 and stride is 2, followed by max-pooling with kernel size of 64 and stride 32. After pooling we apply dropout and project 256-dimensional hidden vector to 2 dimensions by fully-connected layer. The architecture is presented on Fig. 1.

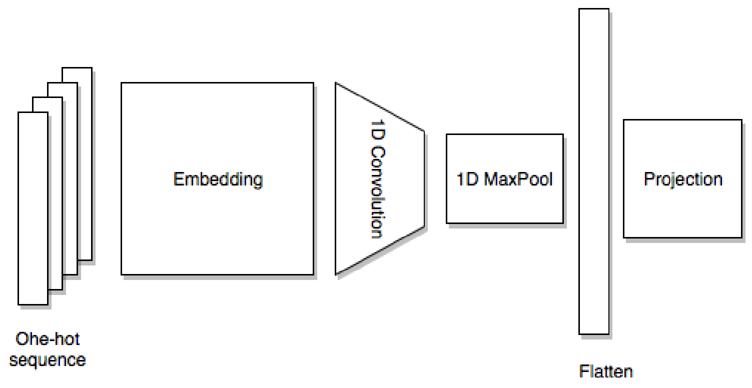

Fig. 1. CharCNN.

3.2 FastText-GRU

The text represented as a sequence 300-dimensional vectors built using pre-trained FastText model. We input this sequence into GRU layer with 256 hidden dim. Next, dropout is applied to the last hidden state and resulting vector is projected into 2 dimension space.

3.3 CharCNN-WordRNN

This model architecture is very similar to [Kim et al., 2016], with exception of highway layer absence. Each word is represented as a sequence of one-hot symbols and the text is represented as sequence of word representations. Words are embedded via convolutional layer with kernel size 5 and max-over-time pooling. The embeddings enter GRU(128) layer, the dropout and projection layers are as described in previous section.

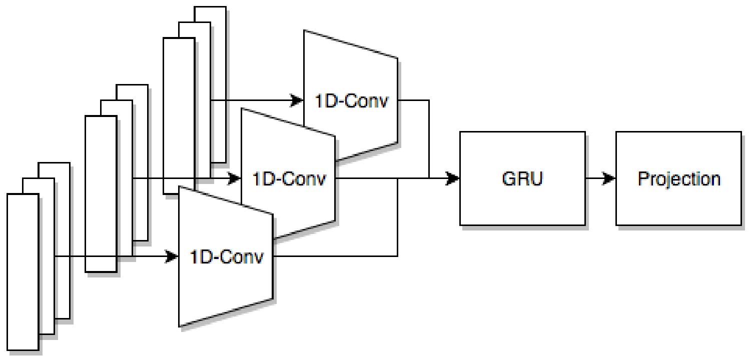

Fig. 2. CharCNN-WordRNN

3.4 CharCNN-WordRNN with attention

This model is referred to as *Attention* below. The model is following to CharCNN-WordRNN, except an attention mechanism [Bahdanau et al., 2014], which is applied to GRU hidden states right after GRU layer. We use attention mechanism analogous to [Vaswani et al., 2017]. Since we are using one so called head in our experiments, it is equivalent to classical attention described in section 2.2.

4 Experiments

4.1 Datasets

We have conducted experiments on two datasets: Movie Review Dataset for English language and Russian Twitter sentiment Dataset for Russian language.

Movie Review Dataset is described in [Maas et al., 2011], is consists of 25,000 positive and 25,000 negative movie reviews form IMDB database. The dataset is evenly divided in train- and testset. The train/test split for the dataset is published.

Russian Twitter Sentiment Dataset was introduced in work [Rubtsova, 2014]. The available version of this dataset consists of 114,991 positive tweets and 111,923 negative tweets, and also large amount of neutral tweets. In our work we use only two classes positive and negative one.

Since there is no published split for this dataset, we created one. We have merged positive and negative sets, shuffled the merged dataset and took first 75% as trainset, next 15% as validation set and the last 15% are treated as test set in our experiments.[1]

One important feature should be mentioned for this dataset: since it was automatically collected by filtering containing emojis (positive and negative ones respectively), these emojis are easy to catch feature for any model, predicting mentioned classes. The emojis in text are short sequences of symbols - from one to three symbols in a row. Such sort sequences are not as easily corrupted by noise, as words themselves, due to typical word length in English is 5.1 letters and in Russian it is 5.28 letters [Bochakrev et al., 2015]. So to complicate the task and research the robustness properties we decided to ignore the punctuation signs related to emojis, namely: ")" and "(".

4.2 Noise Model

In order to demonstrate robustness against noise we have performed spell-checking[2] of described datasets and artificially introduce noise to our datasets. We model:

- the probability of inserting a letter after the current one,
- the probability of a letter to be omitted

for every letter of the input alphabet for each of the task. On each input string we perform random letter insertions and deletions with the modelled probabilities. Both types of noise are added at the same time. We test models with the different levels of noise from 0% (no noise) to 20%. According to [Cucerzan and Brill, 2004], the real level of noise in user-generated texts is 10-15%.

We have conducted three types of experiments:

- the train- and testsets are spell-checked and artificial noise in inserted;
- the train- and testsets are not changed (with above mentioned exception for Russian corpus) and no artificial noise is added;
- the trainset is spell-checked and noised, the testset is unchanged.

These experiment setups are meant to demonstrate the robustness of tested architectures to artificial and natural-born noise.

[1] Authors are going to publish their train/val/test split for the sake of reproducibility.
[2] We have used publicly available enterprise-level spell-check engine Yandex.Speller.

Model	Movie Review	Twitter Sentiment
CharCNN	0.74	0.77
FastTextGRU	0.84	0.76
CharCNN-WordRNN	0.80	0.81
Attention	0.68	0.81

Table 1. Results of the experiments on unchanged dataset. F_1-score on test set.

5 Results

All models are trained with batch size 32, Adam optimizer and dropout before the last fully-connected layer with keep probability 0.5. A loss function is cross-entropy loss. CNN weights are initialized using Xavier initialization with normal distribution.

The results of training and testing models without either spell-checking or artificial noise are presented below.

The results in the Table 1 are presented for reference to compare with the following results in noisy environment.

5.1 Movie Review Dataset

On the Fig. 3 there are presented results in following environment: trainset is cleared from natural noise and introduced the artificial one; testset is also a subject to described transformation. We can see that FastText-GRU model is the best with no presented noise, but it is not that robust. The most durable model is CharCNN-WordRNN one. Surprisingly the Attention model is the worst in this experiment.

On the Fig. 4 there are presented results in following environment: trainset is cleared from natural noise and introduced the artificial one; testset is remained unchanged. The behaviour of all models is roughly the same, but Attention model is demonstrating more robustness on the real test data.

5.2 Russian Twitter Sentiment Dataset

On the Fig. 5 there are presented results in following environment: trainset is cleared from natural noise and introduced the artificial one; testset is also a subject to described transformation. We can spot that the order of tested models is remaining the same. Attention and CharRNN-WordRNN models are really close.

On the Fig. 6 there are presented results in following environment: trainset is cleared from natural noise and introduced the artificial one; testset is remained unchanged. As we can see the demonstrated behaviour of the robustness properties is now more complex. CharCNN and Attention models are performing better on the high noise levels.

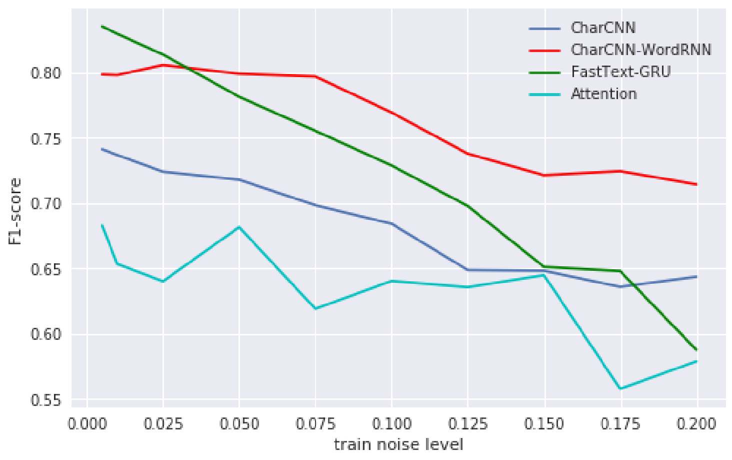

Fig. 3. Movie Review Dataset. Train on spell-checked and noised data, test on spell-checked and noised with the same noise level as train.

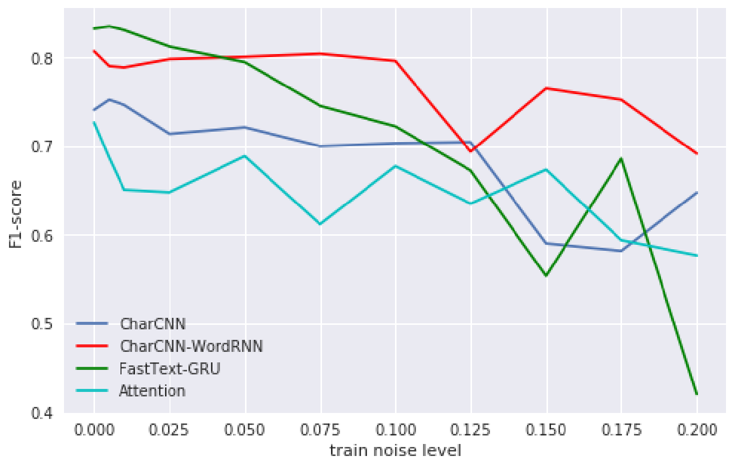

Fig. 4. Moview Review Dataset. Train on spell-checked and noised data, test on unchanged.

6 Conclusion

We have demonstrated the robustness of common modern text classification architectures. And moreover a proposed artificial noise is demonstrated to be adequate surrogate of natural noise in the data. Interestingly some models are performing better on highly noised training data, which could be explained by their internal adaptiveness to proposed noise. The most robust model in the all experiments is the one combining CNN over characters and RNN over words.

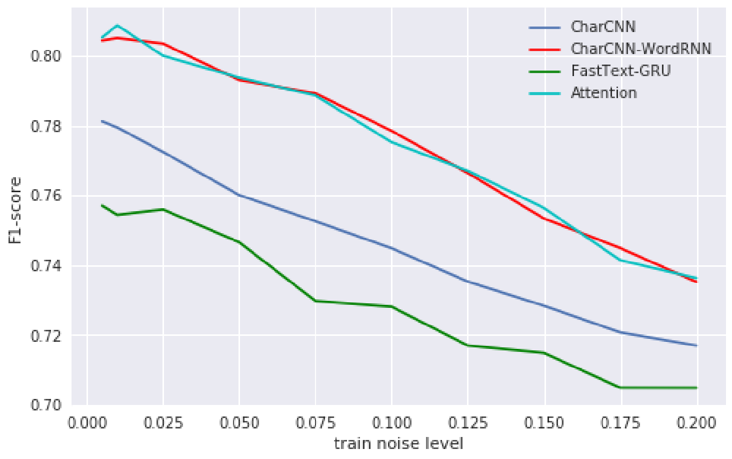

Fig. 5. Twitter Sentiment Dataset. Train on spell-checked and noised data, test on spell-checked and noised with the same noise level as train.

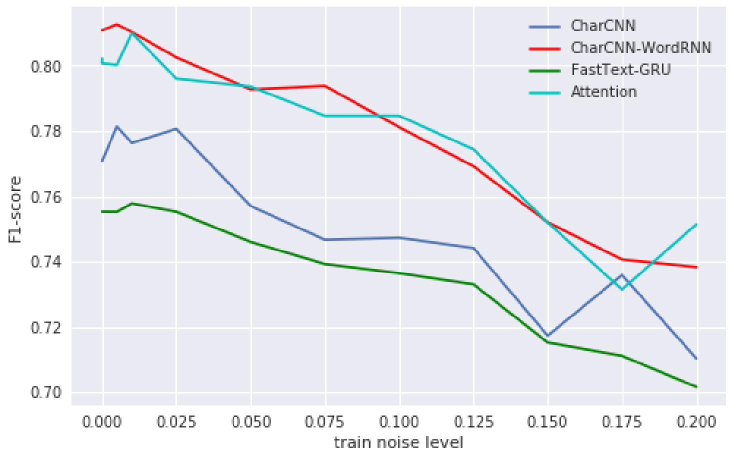

Fig. 6. Twitter Sentiment Dataset. Train on spell-checked and noised data, test on unchanged.

This fact could be explained by a consideration of CNN being insensitive to small changes in the input, RNN being able to capture the meaning of the whole sequence basing on CNN-produced embeddings.

The authors see the future directions for work in this field in three main domains: introduction of other noise types, testing more advanced architectures for text classification and experiments with more complex dataset, containing multi-class and/or multi-label classification.

References

[Joulin et al., 2016] Armand Joulin, Edouard Grave, Piotr Bojanowski and Tomas Mikolov 2016. *Bag of Tricks for Efficient Text Classification.* arXiv preprint arXiv:1607.01759.

[Malykh, 2018] Valentin Malykh 2018. *Robust Word Vectors: Embeddings for Noisy Texts.*

[Kim et al., 2016] Yoon Kim, Yacine Jernite, David Sontag and Alexander M. Rush 2016. *Character-Aware Neural Language Models.* In AAAI, pages 2741–2749.

[Maas et al., 2011] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts 2011. Learning Word Vectors for Sentiment Analysis In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 142–150.

[Rubtsova, 2014] Rubtsova Yuliya 2014. Automatic Term Extraction for Sentiment Classification of Dynamically Updated Text Collections into Three Classes In Knowledge Engineering and the Semantic Web, pp140-149, Springer

[Bochakrev et al., 2015] Bochkarev, V. V., Shevlyakova, A. V., and Solovyev, V. D. 2015. The average word length dynamics as an indicator of cultural changes in society. Social Evolution & History, 14(2), 153-175.

[Cucerzan and Brill, 2004] Cucerzan, S. and Brill, E. 2004. Spelling correction as an iterative process that exploits the collective knowledge of web users. In Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing.

[Joulin et al., 2016] Joulin, A., Grave, E., Bojanowski, P. and Mikolov, T. 2016. Bag of tricks for efficient text classification. arXiv preprint arXiv:1607.01759.

[Howard and Ruder, 2018] Howard, J. and Ruder, S. 2018. Fine-tuned Language Models for Text Classification. arXiv preprint arXiv:1801.06146.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. and Polosukhin, I. 2017. Attention is all you need. In Advances in Neural Information Processing Systems (pp. 6000-6010).

[Zhang et al., 2015] Xiang Zhang, Junbo Jake Zhao and Yann LeCun 2017. Character-level Convolutional Networks for Text Classification. arXiv preprint arXiv:1509.01626

[Kim, 2014] Yoon Kim 2014. Convolutional Neural Networks for Sentence Classification. arXiv preprint arXiv:1408.5882

[Cho et al., 2014] KyungHyun Cho, Bart van Merrienboer, Dzmitry Bahdanau and Yoshua Bengio 2014. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches arXiv preprint arXiv:1409.1259

[Bahdanau et al., 2014] Dzmitry Bahdanau, Kyunghyun Cho and Yoshua Bengio 2014. Neural Machine Translation by Jointly Learning to Align and Translate arXiv preprint arXiv:1409.0473

[Bengio et al., 1994] Bengio, Y., Simard, P. and Frasconi, P. 1994. Learning long-term dependencies with gradient descent is difficult. IEEE transactions on neural networks, 5(2), pp.157-166.