End-To-End Spam Classification With Neural Networks[*]

Christopher Lennan, Bastian Naber, Jan Reher, Leon Weber

1 Introduction

A few years ago, the majority of the internet's network traffic was due to spam (cf. Cormack, 2007a). While we could not locate more recent estimates, the problem of spam definitely persists until today. Spammers have the possibility to adapt to counter-measures such as machine-learning based spam filters. We explore end-to-end deep learning for spam classification because the missing step of feature engineering allows to quickly adapt to the changing tactics of spammers.

2 Background & Related Work

There have been numerous applications of Neural Networks to the problem of spam classification with promising results. To the best of our knowledge all of those models were Multilayer Perceptrons (MLP). Clark et al. (2003) reports that MLPs outperform several other classification methods in terms of accuracy. Similar results are presented by Ruan and Tan (2010) which claims that MLPs with suitable features can outperform several other published methods for spam classification using only a very small number of features.

Convolutional Neural Networks (CNNs) have been shown to perform well on text classification tasks. Kim (2014) uses CNNs for sentence classification and reports state of the art results using very simple architectures. Johnson and Zhang (2015) finds that CNNs work similarly well for text classification. Like most text classification algorithms both represent text on the word level. Recently, there has been increasing interest in representations which exploit the character level structure of words. Zhang et al. (2015) evaluate CNNs using only the stream of characters as input against a huge number of common text classification methods. They find that on suitably large data sets all of those methods are outperformed by the character-level CNNs.

[*]Project report for Machine Leaning 1, SS 2016, Prof. Marius Kloft

3 Methods

We apply two convolutional neural networks for email spam classification, one on character level, the other using word embeddings. As a baseline comparison a character-level support vector machine is implemented as well.

Additionally we experimented with character-level C-LSTM methods inspired by Zhou et al. (2015), which we did not pursue further due to poor preliminary results[1]

3.1 Character-Level Methods

For the character-level quantization of an email we follow the framework by Zhang et al. (2015). That is, from each email only the first n_c characters are considered, where n_c is a predetermined number[2]. Each of the n_c characters is encoded in a one-hot fashion based on an alphabet of length l_a. In our application we used the following alphabet of size 70 (including the new line character):

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

One-hot encoding means that each character is represented by an array of length l_a with 1 at the corresponding alphabet-index and all other elements being 0. For example, the character c would be an array of 0s except for a 1 at the third place. In the end each email is encoded as a $n_c \times l_a$ matrix.

3.1.1 Convolutional Neural Network

We replicate the character-level CNN of Zhang et al. (2015) with 6 convolutional layers, followed by 2 fully connected layers, and a read-out layer. An illustration of the model can be seen in figure 1 and a detailed configuration in table 2 in the Appendix. The 1-D convolutions are applied along the character dimension with varying window sizes. The convolutional filters cover the entire depth of the alphabet, e.g. a filter with window size 7 would be of dimensions 7×70. Each convolution is followed by a max pooling step.

Each fully connected layer is followed by a dropout layer to avoid overfitting (Hinton et al., 2012), with a dropout probability of 50%. The non-linearity in the network is introduced by a thresholding function $h(x) = max\{0, x\}$ at each node, except for the read-out layer, where a softmax function is applied at the two output nodes (binary ham/spam classification).

[1]See section 7.1 in the Appendix for a description of the method.

[2]Zhang et al. (2015) used $n_c = 1014$. In our applications it ranges between 100, 500 and 1000.

3.1.2 Support Vector Machine Baseline

As a baseline comparison we also applied a linear[3] support vector machine (SVM), where the one-hot encoded matrix of each email is flattened to a vector of length $n_c \cdot l_a$.

3.2 Word Embeddings

Word embeddings are dense vector representations of words obtained by projecting tokens into some vector space. The resulting vectors have been shown to capture interesting semantic properties of the represented tokens and to improve performance on NLP tasks when compared with traditional bag-of-words representations (cf. Mikolov et al., 2013). Motivated by those properties, we evaluate a simple CNN architecture which applies convolutions onto embeddings, which in turn are learned by employing a lookup table as described by Collobert and Weston (2007).

The full architecture is as follows (see figure 2): The input is fed into an embedding layer, which is followed by a 1-D convolutional layer. Only the maximum activation of each filter is preserved. Those activations of the convolutional layer are fed into a fully-connected layer. The output of this fully connected layer is then used as input for the final softmax. The embedding layer and the fully connected layer are both regularized by softmax and the L2-norm of the weight vectors.

4 Data & Methodology

We evaluate our methods on two public spam corpora: TREC 2007 and SpamAssassin.

The TREC 2007 public corpus consists in total of 75,419 emails including header. 25,220 (33%) of them are labeled as ham and 50,199 (67%) as spam[4]. All messages were sent to a certain server between April 8 and July 6, 2007. The server included unused email-accounts receiving a lot of spam as well as 'honeypot' accounts, which were published publicly or registered for newsletters and other services (see Cormack, 2007b, p. 3f.).

The SpamAssassin public mail corpus consists of 1,897 spam and 4,150 ham messages[5].

For the evaluation we split each of the two data sets into a training set (75% of the data) and a test set (remaining 25%). Hyperparameters are optimized on the training set and the best model is evaluated on the held-out test set.

[3]A SVM with Gaussian kernel was also tested resulting in a slightly worse accuracy score and a significantly higher run time

[4]Further information and data download are available at `http://plg.uwaterloo.ca/~gvcormac/treccorpus07/about.html`

[5]Information and download at `https://spamassassin.apache.org/publiccorpus/`

Table 1: Accuracy scores on the two data sets

	TREC 2007	SpamAssassin
SVM (character level)	**1**	0.96
CNN (character-level)	**1**	0.97
CNN (word embeddings)	**1**	**0.98**

Since nested cross-validation is too costly for the CNN framework, we used the same train-test split for all the methods to achieve some comparability.

Both character-level methods use the first 100 characters ($n_c = 100$) on the TREC 2007 data. On SpamAssassin two models with $n_c = 500$ and $n_c = 1000$ are applied (see table 2 and 3 for an overview).

5 Results

In table 1 we report the accuracy scores of our models on the two data set rounded to the second decimal place. For the character-level CNN there is some indication of numerical instability. Therefore the results of that model should be treated with caution.

All methods perform exceptionally well on the TREC 2007 data set. Our analysis indicates a special discriminatory power of the sender's email address, which might be caused by most ham messages in the corpus coming from newsletters and subscribed services that send many mails from the same or similar addresses[6]. The high accuracy score could therefore be an artifact of the data.

On the SpamAssassin data, the CNN using word embeddings outperforms the other models by a small margin. Interestingly, a character-level linear SVM obtains similar accuracy than the more sophisticated other methods. We hypothesize that this is due to the small size of the data set.

6 Conclusion

Since both implemented convolutional neural networks beat our baseline linear SVM in accuracy, the usage of CNNs for spam detection seems promising. Nevertheless, a character-level linear SVM application turned out to be surprisingly powerful, with lower complexity but almost comparable accuracy. For a final verdict about the CNN's applicability to spam classification future research on larger spam data sets that are more representative of real life email traffic is needed.

[6]Similarly, Grendár et al. (2011, p. 6) found that the headers in TREC 2007 contain more substantive information for discriminating between spam and ham than emails of normally used accounts.

Bibliography

Clark, J., I. Koprinska, and J. Poon (2003). A neural network based approach to automated e-mail classification. *Proceedings - IEEE/WIC International Conference on Web Intelligence, WI 2003*, 702–705.

Collobert, R. and J. Weston (2007). Fast semantic extraction using a novel neural network architecture. In *Annual meeting-association for computational linguistics*, Volume 45, pp. 560.

Cormack, G. V. (2007a). Email spam filtering: A systematic review. *Foundations and Trends in Information Retrieval 1*(4), 335–455.

Cormack, G. V. (2007b). Trec 2007 spam track overview. In *Sixteenth Text REtrieval Conference (TREC-2007)*, Gaithersburg, MD.

Grendár, M., J. Škutová, and V. Špitalskỳ (2011). Spam filtering by quantitative profiles. *arXiv preprint arXiv:1201.0040*.

Hinton, G. E., N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov (2012). Improving neural networks by preventing co-adaptation of feature detectors. *CoRR abs/1207.0580.*

Hochreiter, S. and J. Schmidhuber (1997). Long short-term memory. *Neural computation 9*(8), 1735–1780.

Johnson, R. and T. Zhang (2015). Effective Use of Word Order for Text Categorization with Convolutional Neural Networks. *Naacl* (2011), 103–112.

Kim, Y. (2014). Convolutional Neural Networks for Sentence Classification. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, 1746–1751.

Mikolov, T., G. Corrado, K. Chen, and J. Dean (2013). Efficient Estimation of Word Representations in Vector Space. *Proceedings of the International Conference on Learning Representations (ICLR 2013).*

Ruan, G. and Y. Tan (2010). A three-layer back-propagation neural network for spam detection using artificial immune concentration. *Soft Computing 14*(2), 139–150.

Zhang, X., J. Zhao, and Y. LeCun (2015). Character-level convolutional networks for text classification. *arXiv preprint arXiv:1509.01626*.

Zhou, C., C. Sun, Z. Liu, and F. Lau (2015). A c-lstm neural network for text classification. *arXiv preprint arXiv:1511.08630*.

7 Appendix

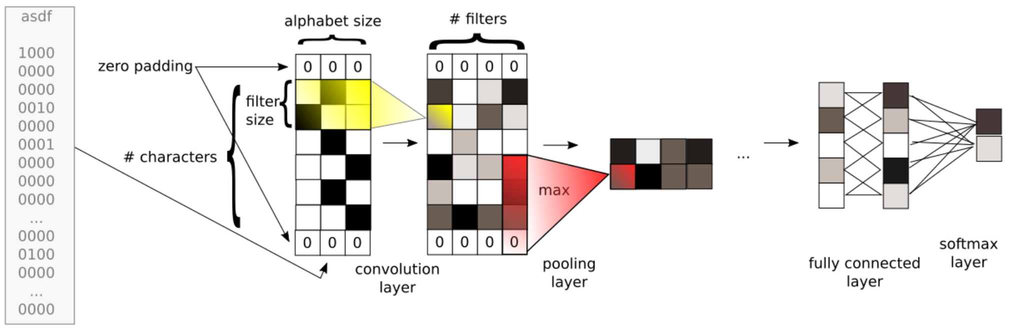

Figure 1: Architecture of the character-level CNN

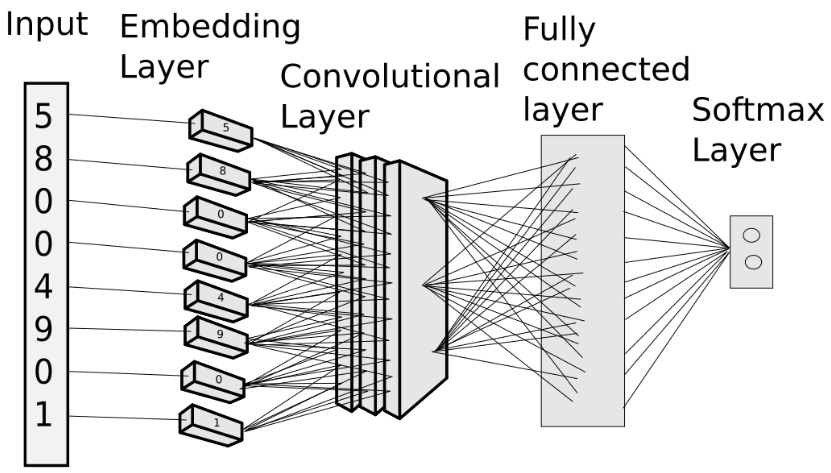

Figure 2: Architecture of the CNN using embeddings

7.1 C-LSTM

Recurrent Neural Networks (RNN) are a powerful model to learn sequential correlations in the input data and have achieved state of the art performance in many problems using sequential data such as speech recognition or translation. Due to the vanishing and exploding gradient problem, several methods have been developed to allow the memory component of a RNN to better incoorporate long-term dependencies in the data. One of the more successful methods is the Long-Short-term Memory proposed by Hochreiter and Schmidhuber (1997). A LSTM cell contains a memory state and several gating mechanisms that determine how the memory state is to be updated based on the current input.

Since CNNs are excellent feature extractors, it is reasonable to combine the features learned by CNNs with LSTMs for classification purposes. Drawing inspiration from Zhou et al. (2015), we train a C-LSTM on the Spam Assassin dataset. In contrast to Zhou et al. (2015), however, who trained their network on word embeddings at the sentence level, we train the network on character level on the whole document. As in our CNN, we one-hot encode the input using the same alphabet, apply one layer of filters and treat the feature maps from the convolutional layer as input to the LSTM. The memory state component after the last input in the sequence is used as the document representation and is used as input to a Softmax layer. The network is trained by minimizing the cross entropy error using the RMSProp algorithm.

Table 2: CNN models overview

		Zhang et. al.	CNN 1	CNN 2	CNN 3
1st convolutional layer	number of filters	256 / 1024	256	256	256
	window size	7	7	7	7
	max. pooling factor	3	3	3	3
2nd convolutional layer	number of filters	256 / 1024	256	256	256
	window size	7	7	7	7
	max. pooling factor	3	3	3	3
3rd convolutional layer	number of filters	256 / 1024	256	256	256
	window size	3	3	3	3
	max. pooling factor	1	1	1	1
4th convolutional layer	number of filters	256 / 1024	256	256	256
	window size	3	3	3	3
	max. pooling factor	1	1	1	1
5th convolutional layer	number of filters	256 / 1024	256	256	256
	window size	3	3	3	3
	max. pooling factor	1	1	1	1
6th convolutional layer	number of filters	256 / 1024	256	256	256
	window size	3	3	3	3
	max. pooling factor	3	3	3	3
1st fully connected layer	output units	1024 / 2048	1024	1024	1024
	dropout probability	50%	50%	50%	50%
2nd fully connected layer	output units	1024 / 2048	1024	1024	1024
	dropout probability	50%	50%	50%	50%
Read-out layer	output units	depends on task	2	2	2
Batch size			75	75	75
Learning rate			0.0005	0.0005	0.0006
Number of characters		1014	100	500	1000
Accuracy	TREC 2007		99.8%		
	Spam assassin		95.3%	96.5%	97.2%

Hyperparameters batch size and learning rate are optimized via 8-fold cross validation on the train set

Table 3: Character-Level SVM-Models

	Kernel	n_c	C	TREC 2007	SpamAssassin
SVM 1	linear	100	0.1	99.83%	-
SVM 2	linear	500	0.01	-	95.44%
SVM 3	linear	1000	0.01	-	95.77%

SVM applications with optimal hyperparameter C and accuracy score on the two data sets

n_c = number of characters, C = regularization constant

C is optimized via grid search on the train set evaluated by 5-fold cross validation.