

# Supervised Spam Classification

“Comparing Effectivity and Robustness of Sequential and Non-Sequential Machine-Learning Models”

Atreya Shankar, Cognitive Systems (M.Sc.)

BM2: Intelligent Data Analysis (IDA)

University of Potsdam, SoSe 2019

Prof. Dr. Tobias Scheffer

August 08. 2019

# Table of Contents

1 Introduction

2 Methodologies

3 Results

4 Evaluation

5 Conclusions

6 Bibliography

# Introduction

## Intelligent Data Analysis

Exam: Spam (Project 5)

Summer term 2018

Prof. Tobias Scheffer  
Dr. Paul Prasse  
Silvia Makowski  
Dr. Lena Jäger

This project is part of the exam *Intelligent Data Analysis*. Each project assignment is to be resolved by a single student on his/her own. The student is supposed to present the solution as part of the oral exam. The student is required to present a printed version of the Python code together with diagrams, tables, etc. that summarize the results. The specific way of how the project is presented is up to the student's choice.

### Problem setting

You have been hired by the IT department of a medium-sized company to train an email spam filter which should mark the incoming emails of all employees as spam or non-spam. The emails are parsed by a module and converted into the bag-of-words representation. A total of 57,173 different words (features) are distinguished. The aim of the filter is to identify a maximum number of spam emails, with a maximum of 0.2% of all legitimate emails being classified incorrectly. In addition, the company wants to make a statement about the effectiveness of the filter on future emails, i.e., what percentage of incoming spam emails will be identified in the future.

### Aufgabe

From the employees' inboxes, 10,000 emails were extracted as training data (see emails.mat). Let  $X$  be the training data with the associated class labels  $Y$  (+1 stands for *spam*, -1 means *non-spam*). Identify a suitable learning technique for constructing a spam filter and implement it in Python. Train and evaluate the model. Make a statement about the expected quality of the filter and make sure that no more than 0.2% of all legitimate emails are filtered. For this purpose, plot a precision/recall curve and mark the model's position on the curve (for the selected threshold). Briefly motivate and document all the steps you have taken.

Fig. 1. Spam project description

- Project description proposes using data in “emails.mat” file with 10k instances and ~50k features
- Bag-of-words form of data, which would only work for non-sequential learning
- Enron-spam pre-processed text data derived from Enron Corporation scandal; subset of employees' emails became publicly available (Metsis, Androutsopoulos, and Paliouras, 2006)
- Consists of 33,716 text-based emails; 16,545 “ham” and 17,171 spam instances

# Objectives

- Utilize enron-spam emails database to implement both sequential and non-sequential supervised classifiers
- Meet project requirement to develop a classifier that attains 99.8% recall on “ham” emails
- Provide input into recall values for future spam emails given selected optimal threshold
- Additionally, provide insights into effectivity and robustness of sequential and non-sequential models

# Overview

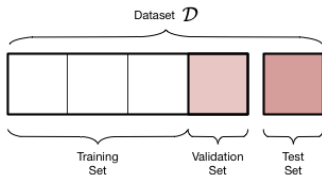


Fig. 2. Data splitting schematic (Ziganto, 2018)

- Non-sequential model: Support Vector Machine (SVM)
- Sequential model: CNN-LSTM with word/character embeddings
- Due to time limitations, K-fold cross-validation was omitted
- Compromise: train/validate/test on the same subsets of data for fair comparison
- $(\text{Train} \cup \text{Validation}):\text{Test} \implies 70:30$
- $\text{Train}:\text{Validation} \implies 85:15$

# Non-Sequential Model: Support Vector Machine (SVM)

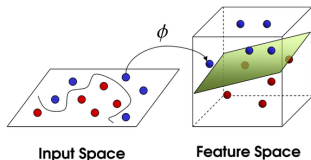


Fig. 3. Support Vector Machine (SVM) schematic (Joshi, 2012)

- Pre-processing text to normalized bag-of-words representation with  $|V| = 5,000$  words
- Sklearn's `SGDClassifier` with Mini-Batch SGD and early stopping
- Linear and approximated RBF Kernel (`RBFSampler`)
- Grid-search over batch-size, regularization term  $\alpha$ , RBF kernel  $\gamma$ , and number of sampling components for `RBFSampler`

# Sequential Model: CNN-LSTM (Words)

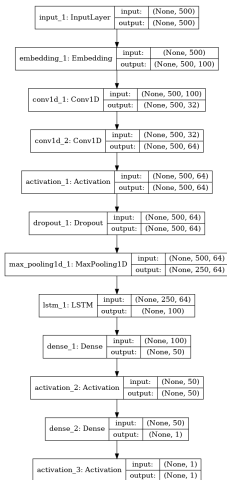


Fig. 4. Keras schematic for CNN-LSTM (Words)

- Pre-processing text to padded/clipped integer encoded tokens with  $|V| = 5,000$  words
- 1-dimensional CNN with varying filters to enrich sequential features; LSTM cell to capture short and long-term sequential relationships; dropout regularization for model robustness
- Grid-search over embedding dimensions, dropout rate, batch-size and learning rate
- Learning both with and without pre-trained GloVe word vectors ( $\sim 6$  billion tokens)

# Sequential Model: CNN-LSTM (Words+Characters)

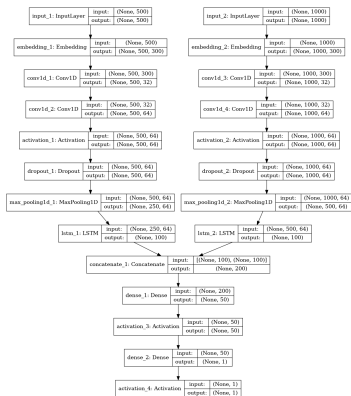


Fig. 5. Keras schematic for CNN-LSTM (Words+Characters)

- Using character sequences to overcome unknown token issue; same general architecture as before
- Grid-search over embedding dimensions, dropout rate, batch-size and learning rate
- Learning both with and without pre-trained GloVe word vectors (~6 billion tokens)
- Approximating GloVe character embeddings by averaging over character-containing word vectors (Woolf, 2017)



# Grid-search optimal models

Classifier	Test $F_1$	ROC-AUC
SVM (Linear Kernel)	0.9836	<b>0.9965</b>
SVM (Approximated RBF Kernel)	0.3437	0.4063
CNN-LSTM (Words)	0.9753	0.9972
CNN-LSTM (Words+Characters)	0.9808	0.9975
CNN-LSTM (Words+GloVe)	0.9902	<b>0.9989</b>
CNN-LSTM (Words+Characters+GloVe)	0.9902	<b>0.9989</b>

**Table 1:** Summary of grid-search optimal models; zero rule classifier baseline is 50.9%;  $F_1$  scores with fixed threshold at 0 and 0.5 for SVM and CNN-LSTM respectively

- Both sequential and non-sequential models achieve high  $F_1$  and ROC-AUC test scores

# “Ham” Relative Importance Analysis (SVM)

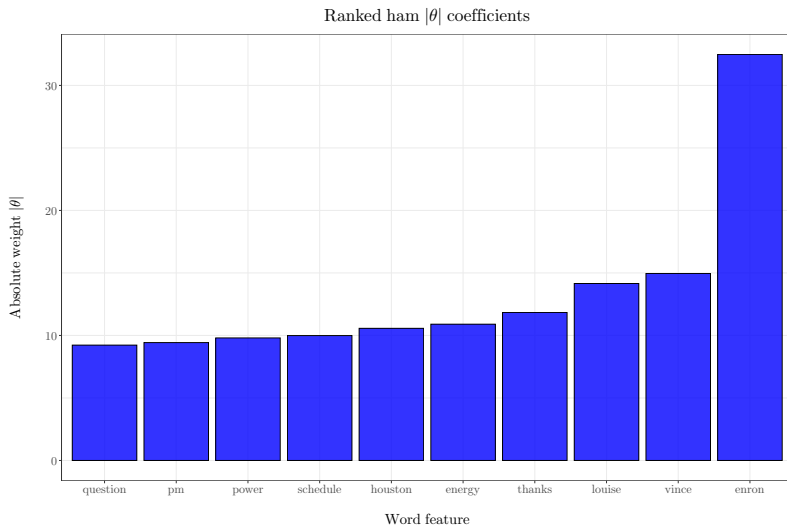


Fig. 6. Relative importance analysis for SVM (linear kernel)

# Spam Relative Importance Analysis (SVM)

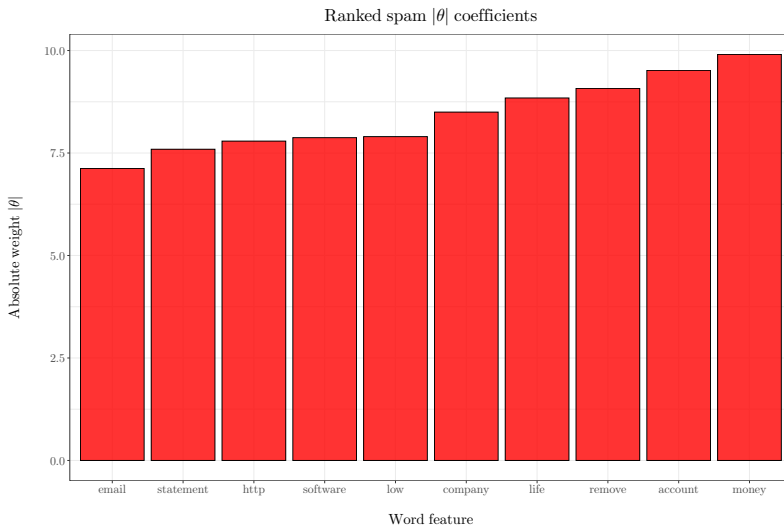


Fig. 7. Relative importance analysis for SVM (linear kernel)

# Optimal Threshold Analysis

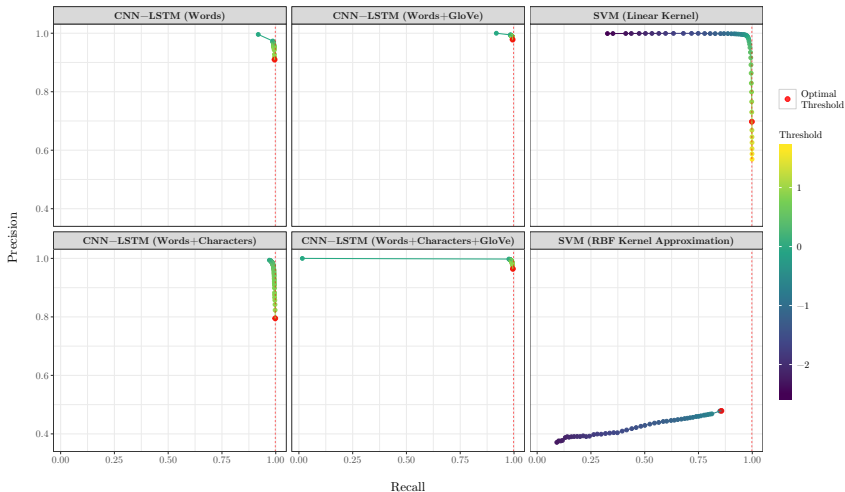


Fig. 8. Precision-Recall curve (ham label) for optimal threshold analysis

# Optimal Threshold Performance

Classifier	Threshold	Recall [Spam]	Recall [Ham]
SVM (Linear Kernel)	1.171	0.5509	0.9982
SVM (Approximated RBF Kernel)	5.552	0.1558	0.8991
CNN-LSTM (Words)	0.9444	0.9224	0.9943
CNN-LSTM (Words+Characters)	0.9444	0.8137	0.9971
CNN-LSTM (Words+GloVe)	0.9444	0.9846	0.9929
CNN-LSTM (Words+Characters+GloVe)	0.9444	0.9781	0.9930

Table 2: Results of optimal threshold analysis

- Trade off between ham and spam recall; either accept low recall for spam or compromise with lower recall for ham

# Blind Dataset Performance (SMS Spam)

Classifier	Blind $F_1$	ROC-AUC
SVM (Linear Kernel)	0.4688	<b>0.7039</b>
SVM (Approximated RBF Kernel)	0.1785	0.4937
CNN-LSTM (Words)	0.5090	0.6158
CNN-LSTM (Words+Characters)	0.4416	0.6522
CNN-LSTM (Words+GloVe)	0.2913	<b>0.7567</b>
CNN-LSTM (Words+Characters+GloVe)	0.3017	<b>0.7578</b>

**Table 3:** Results of blind data test; zero rule classifier obtains 87% due to class imbalance;  $F_1$  scores with fixed threshold at 0 and 0.5 for SVM and CNN-LSTM respectively

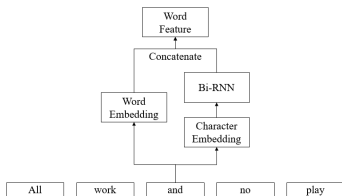
- Words-based models perform consistently well on blind dataset (albeit worse than zero rule classifier)
- Considering sequential nature of data contributes to some robustness

# Conclusions

- For spam detection, both sequential and non-sequential models are effective
- Trade-off exists between “ham” and spam recall; an informed decision must be made
- Both words-based sequential and non-sequential models tend to be robust to new datasets; although sequential models tend to carry richer and more discriminating features
- High cost of training CNN-LSTM; perhaps not economical for a company to deploy GPU on IMAP server
- **SVM would be a more efficient and scalable option**

# Improvements to Embeddings in CNN-LSTM

- Separate pipeline for character sequences leads to symbolic overfitting on types of datasets
- Can overcome unknown tokens but contributes uncertainty in terms of dialects and expressions
- Zhao (2018) proposes a bidirectional LSTM to enrich word vector features
- This could address the unknown token issue without leading to overfitting on entire character sequences



**Fig. 9.** Improved word-character embedding model (Zhao, 2018)



# Bibliography I

Joshi, Prateek (2012). URL:

<https://prateekvjoshi.com/2012/08/24/support-vector-machines/>.

Metsis, Vangelis, Ion Androutsopoulos, and Georgios Paliouras (2006). "Spam filtering with naive bayes-which naive bayes?" In: *CEAS*. Vol. 17. Mountain View, CA, pp. 28–69.

Woolf, Max (2017). *char-embeddings (Github)*. URL:

<https://github.com/minimaxir/char-embeddings>.

Zhao, HG (2018). *keras-word-char-embs (Github)*. URL:

<https://github.com/CyberZHG/keras-word-char-embd>.

# Bibliography II

Ziganto, David (2018). *Model Tuning (Part 2 - Validation & Cross-Validation)*. URL:

<https://dziganto.github.io/cross-validation/data%20science/machine%20learning/model%20tuning/python/Model-Tuning-with-Validation-and-Cross-Validation/>.