# Hyperpartisian News detection

Florian Biebel    Daan Middendorp

July 10, 2019

# Table of contents

Introduction

Different basic attemps

- Pipeline
- Article measures
- Scores

CNN-Approach

Conclusion

# Introduction

- Hyperpartisian News
  - "Extremely biased in favor of a political party"
- SemEval 2019 competition

# Datasets

1. byarticle ($\sim$ 600 articles directly labeled, 2,7MB; *quality data*)
2. bypublisher ($\sim$ $600k$ articles indirectly labeled by publisher, 2,6GB; *quantity data*)
3. byart-bypub-mix (byarticle $+$ 1100 articles from bypublisher)

due to time and ressource constraints, only byarticle and mix were used, as well as a seed

# Attempts

- Classifiers
  - Logistic Regression
  - SGD Classifier
  - Random Forest Classifier
  - Naive Base Classifier
  - CNN Keras
- Feature Unions
  - Length of article
  - Number of capitalized words
  - Number of exclamation marks

# Pipeline

- XML-parser
- Transformer
  - CountVectorizer
  - TfidfTransformer
- Bag of Words
- Classifier

# Article measures

- Hyperpartisian
  - Average of 596 words per article
  - Average of 7 capitalized words per article
  - Average of 0.98 exclamation marks per article
- Non-Hyperpartisian
  - Average of 415 words per article
  - Average of 5 capitalized words per article
  - Average of 0.36 exclamation marks per article

- How to tokenize?

# Measure tokenization

- Add as word
  - LONGARTICLE
  - CAPITALUSAGE
  - EXCLAMATIONMARK
- Feature Union
  - Concatenates results of multiple transformer objects.

# Scores (20 runs)

| Method | Acc. | Prec. | Rec. | F1 |
|---|---|---|---|---|
| Logistic regression | 0,70-0,82 | 0,65-0,81 | 0,38-0,60 | 0,51-0,70 |
| SGD Classifier | 0,62-0,76 | 0,51-0,79 | 0,52-0,78 | 0,59-0,70 |
| Random Forest | 0,63-0,78 | 0,60-0,88 | 0,27-0,47 | 0,41-0,59 |
| Naive Bayes | 0,65-0,78 | 0,50-0,75 | 0,57-0,80 | 0,59-0,71 |
| **Log. reg. FU** | **0,71-0,82** | **0,67-0,92** | **0,39-0,64** | **0,49-0,70** |
| SGD FU | 0,34-0,70 | 0,00-0,40 | 0,00-1,00 | 0,0-0,58 |

# Baselines

- Logistic Regression
- SGDClassifier (SVM)

# Logistic Regression

| | Quality | Mix |
|---|---|---|
| Accuracy | 80 | 68 |
| Precision | 76 | 67 |
| Recall | 60 | 50 |
| F1 | 67 | 57 |

consistent results due to the seed

# SVM

| | Quality | Mix |
|---|---|---|
| Accuracy | 71-76 | 63-69 |
| Precision | 55-64 | 56-60 |
| Recall | 65-80 | 59-70 |
| F1 | 60-66 | 59-65 |

inconsistent results despite seed

due to Stochastic Gradient Descent

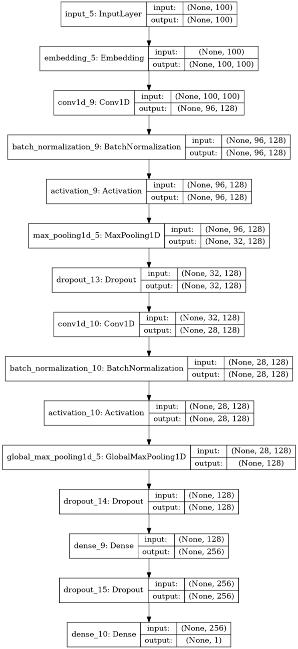
# CNN

|  | Quality | Mix |
|---|---|---|
| Accuracy | 69-74 | 59-66 |
| Precision | 53-63 | 51-61 |
| Recall | 47-67 | 57-72 |
| F1 | 52-60 | 57-61 |

# Techniques

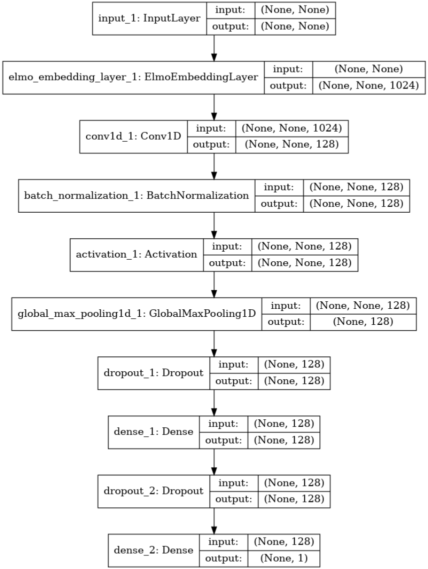
- BatchNormalization
- Dropout
- (Global)MaxPooling
- Learning Embeddings and Pre-Learned Embeddings (ELMo/Word2Vec/GloVe)

# CNN Graph



| | | |
|---|---|---|
| input_5: InputLayer | input: | (None, 100) |
| | output: | (None, 100) |
| embedding_5: Embedding | input: | (None, 100) |
| | output: | (None, 100, 100) |
| conv1d_9: Conv1D | input: | (None, 100, 100) |
| | output: | (None, 96, 128) |
| batch_normalization_9: BatchNormalization | input: | (None, 96, 128) |
| | output: | (None, 96, 128) |
| activation_9: Activation | input: | (None, 96, 128) |
| | output: | (None, 96, 128) |
| max_pooling1d_5: MaxPooling1D | input: | (None, 96, 128) |
| | output: | (None, 32, 128) |
| dropout_13: Dropout | input: | (None, 32, 128) |
| | output: | (None, 32, 128) |
| conv1d_10: Conv1D | input: | (None, 32, 128) |
| | output: | (None, 28, 128) |
| batch_normalization_10: BatchNormalization | input: | (None, 28, 128) |
| | output: | (None, 28, 128) |
| activation_10: Activation | input: | (None, 28, 128) |
| | output: | (None, 28, 128) |
| global_max_pooling1d_5: GlobalMaxPooling1D | input: | (None, 28, 128) |
| | output: | (None, 128) |
| dropout_14: Dropout | input: | (None, 128) |
| | output: | (None, 128) |
| dense_9: Dense | input: | (None, 128) |
| | output: | (None, 256) |
| dropout_15: Dropout | input: | (None, 256) |
| | output: | (None, 256) |
| dense_10: Dense | input: | (None, 256) |
| | output: | (None, 1) |

# ELMo Model



| | | |
|---|---|---|
| input_1: InputLayer | input: | (None, None) |
| | output: | (None, None) |
| elmo_embedding_layer_1: ElmoEmbeddingLayer | input: | (None, None) |
| | output: | (None, None, 1024) |
| conv1d_1: Conv1D | input: | (None, None, 1024) |
| | output: | (None, None, 128) |
| batch_normalization_1: BatchNormalization | input: | (None, None, 128) |
| | output: | (None, None, 128) |
| activation_1: Activation | input: | (None, None, 128) |
| | output: | (None, None, 128) |
| global_max_pooling1d_1: GlobalMaxPooling1D | input: | (None, None, 128) |
| | output: | (None, 128) |
| dropout_1: Dropout | input: | (None, 128) |
| | output: | (None, 128) |
| dense_1: Dense | input: | (None, 128) |
| | output: | (None, 128) |
| dropout_2: Dropout | input: | (None, 128) |
| | output: | (None, 128) |
| dense_2: Dense | input: | (None, 128) |
| | output: | (None, 1) |

# Conclusion

- Logistic regression seems to be the best baseline
- Small improvements possible adding extra features
- Bypublisher dataset makes it worse
- It is hard to apply CNN's, not as easy as thought
- Great resources are needed, also for small machinelearning tasks