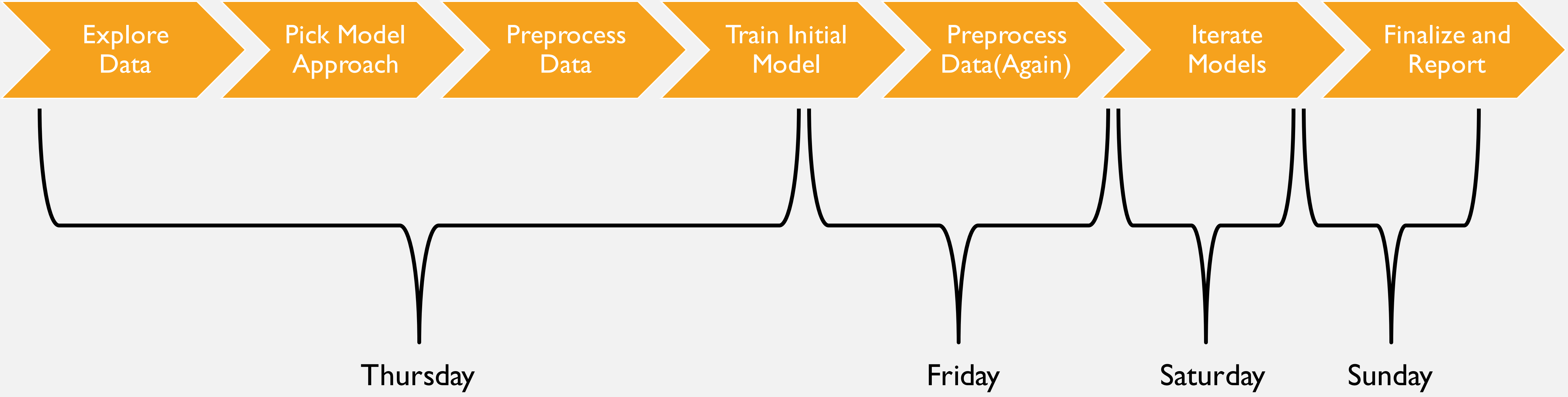
PRODUCT COMPLAINT CLASSIFICATION

How to Achieve Decent Accuracy with Minimal Feature Engineering

EXECUTIVE SUMMARY

- Classify Free-form Complaints
- Using Little Feature Engineering
- Using Minimal Supervised Data
- Achieving ~80% Accuracy

TIMELINE/ANALYTICAL PROCESS



DATA EXPLORATION

- Goal: understand the data set, underlying distributions and features of each text field
- At runtime: 268,380 distinct complaints returned

Product Classification	Number of Complaints
Bank_service	20,071
Credit_card	29,553
Credit_reporting	81,234
Debt_collection	61,471
Loan	31,036
Money_transfers	4,734
Mortgage	40,281

product_group	min_len	mean_len	median_len	max_len	sdev	dist_skew
bank_service	9	1301.404016	969	23066	1180.857376	3.463401
credit_card	14	1204.096741	904	31047	1127.324130	4.801937
credit_reporting	11	836.684196	572	21669	923.101751	4.478349
debt_collection	5	868.183225	586	31735	978.320124	5.910101
loan	8	1214.070982	886	30892	1183.954083	4.657901
money_transfers	17	1173.546050	838	31234	1290.673558	6.587759
mortgage	13	1585.213475	1177	31463	1492.512509	4.980576

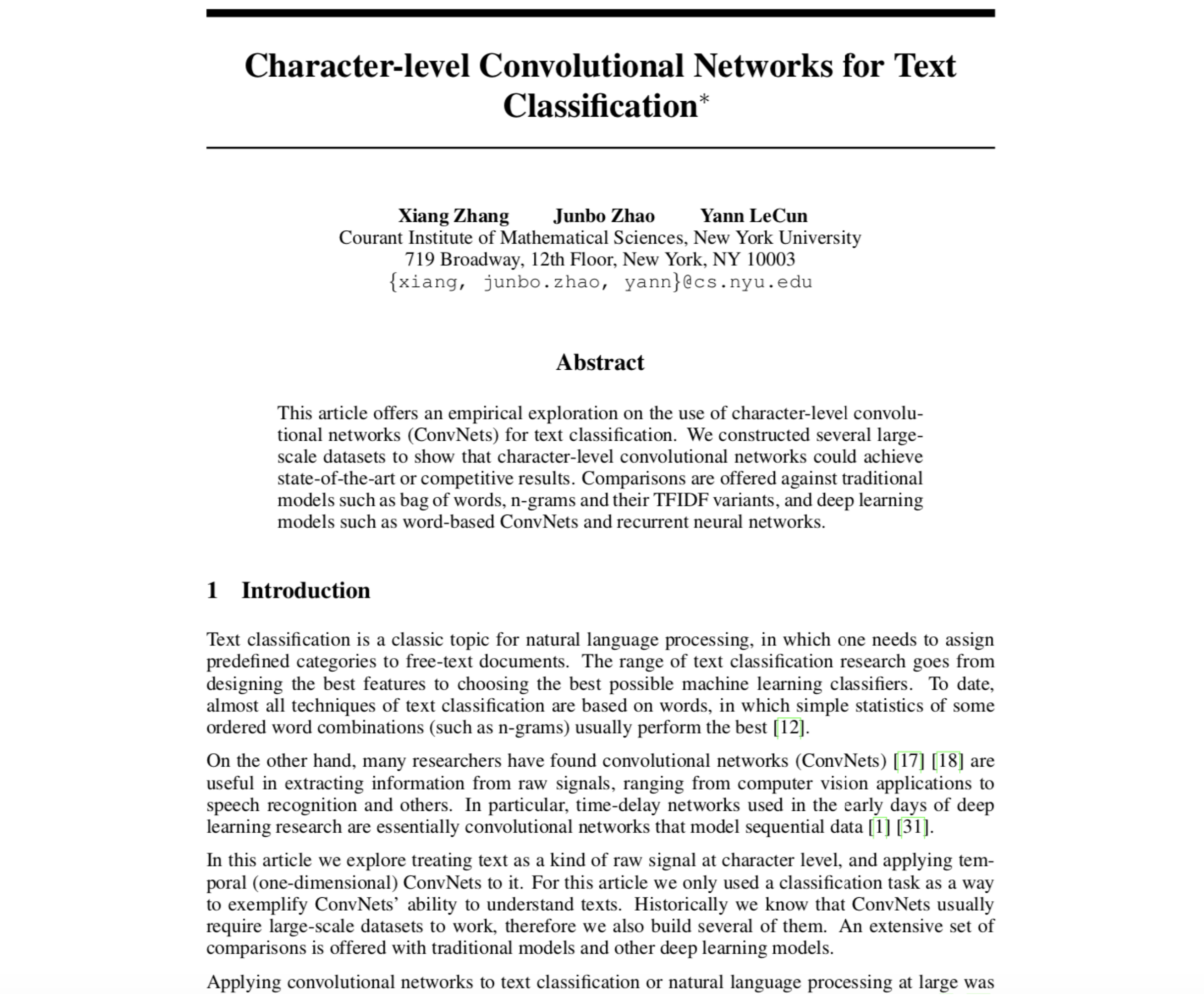
DATA EXPLORATION

Goal: understand distribution of text length per category

PICKING MODEL APPROACH

Wants:

- Easily can extend to new data
- Little feature engineering
- Demonstrates an interesting approach
- Has some comparable benchmarks



Character-level Convolutional Networks for Text Classification*

Xiang Zhang **Junbo Zhao** **Yann LeCun**
Courant Institute of Mathematical Sciences, New York University
719 Broadway, 12th Floor, New York, NY 10003
{xiang, junbo.zhao, yann}@cs.nyu.edu

Abstract

This article offers an empirical exploration on the use of character-level convolutional networks (ConvNets) for text classification. We constructed several large-scale datasets to show that character-level convolutional networks could achieve state-of-the-art or competitive results. Comparisons are offered against traditional models such as bag of words, n-grams and their TFIDF variants, and deep learning models such as word-based ConvNets and recurrent neural networks.

1 Introduction

Text classification is a classic topic for natural language processing, in which one needs to assign predefined categories to free-text documents. The range of text classification research goes from designing the best features to choosing the best possible machine learning classifiers. To date, almost all techniques of text classification are based on words, in which simple statistics of some ordered word combinations (such as n-grams) usually perform the best [12].

On the other hand, many researchers have found convolutional networks (ConvNets) [17] [18] are useful in extracting information from raw signals, ranging from computer vision applications to speech recognition and others. In particular, time-delay networks used in the early days of deep learning research are essentially convolutional networks that model sequential data [1] [31].

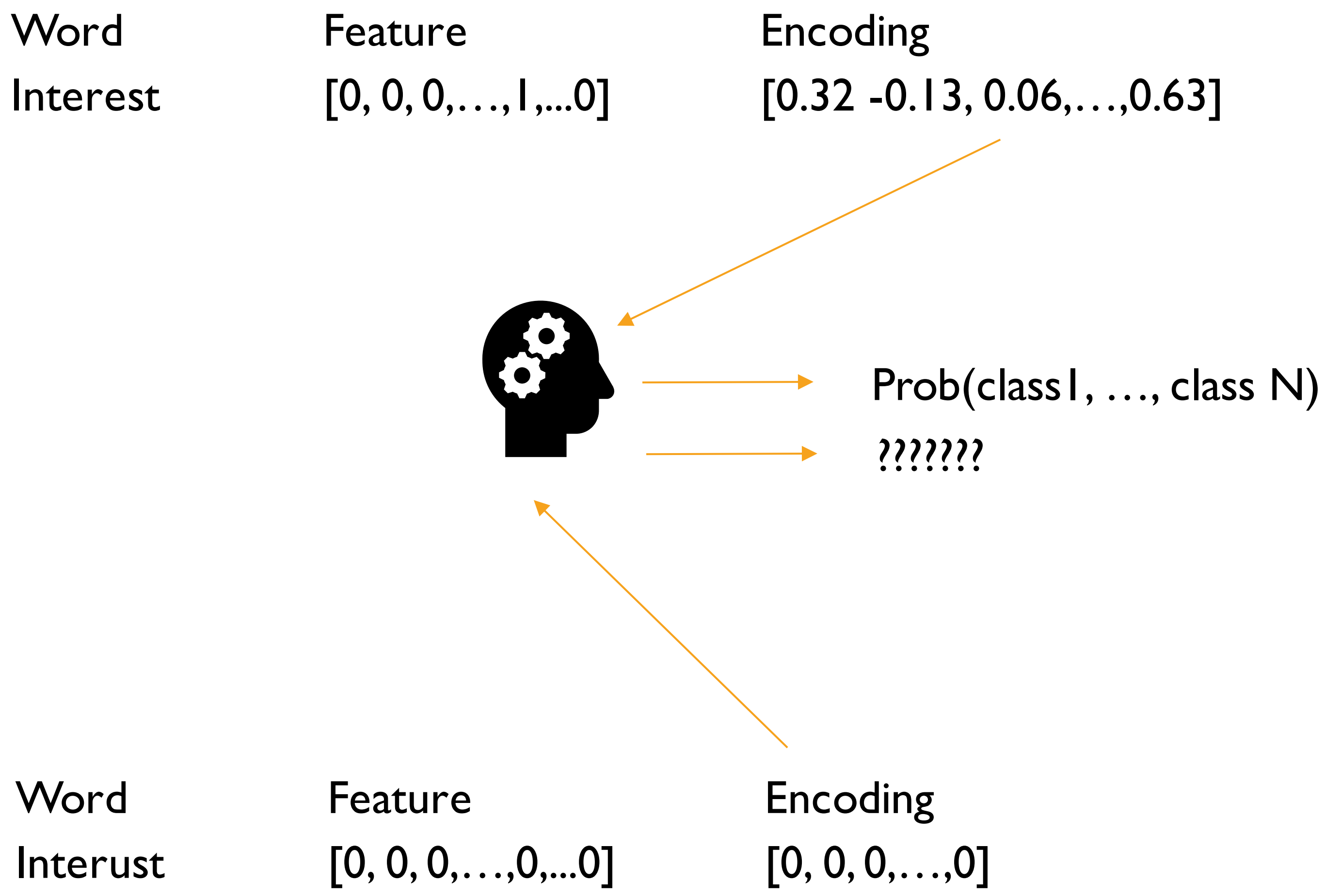
In this article we explore treating text as a kind of raw signal at character level, and applying temporal (one-dimensional) ConvNets to it. For this article we only used a classification task as a way to exemplify ConvNets' ability to understand texts. Historically we know that ConvNets usually require large-scale datasets to work, therefore we also build several of them. An extensive set of comparisons is offered with traditional models and other deep learning models.

Applying convolutional networks to text classification or natural language processing at large was

"FEATURE" ENGINEERING

Building blocks:

- Word counts
- TFxIDF
- Word2vec
- N-grams

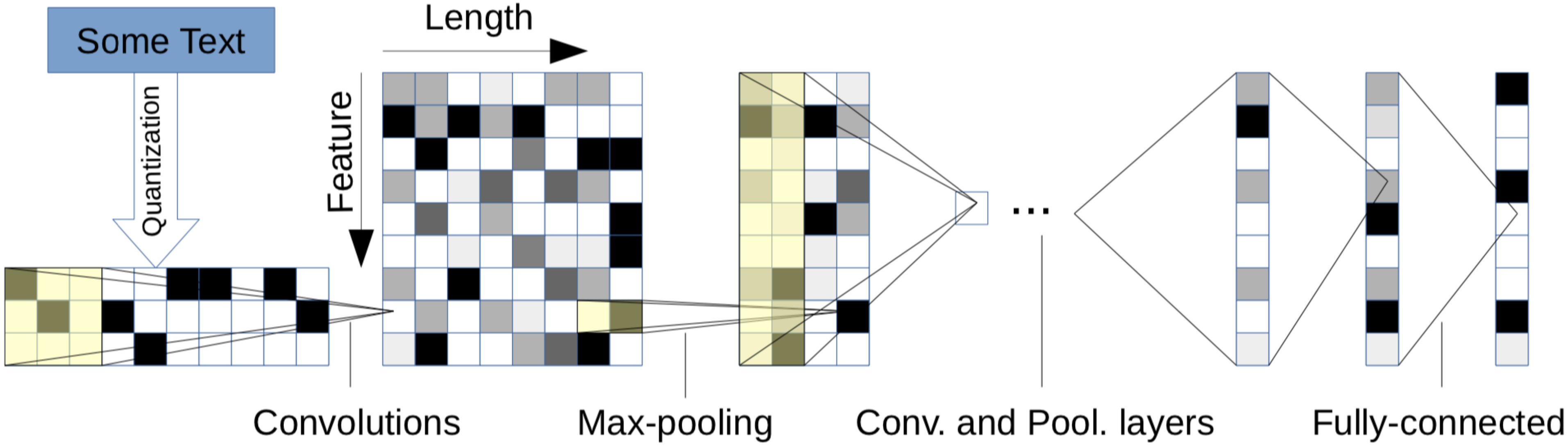


"FEATURE" ENGINEERING

Building blocks:

- Use characters instead!
- A-z
- 0-9
- Symbols as needed

THE MODEL



Some Text
Quantization
Length
Feature
...
Convolutions
Max-pooling
Conv. and Pool. layers
Fully-connected

MODEL AS CODE

```python
def forward(self, _input):
    layer1 = self.conv1(_input)
    layer1 = self.conv1_bn(layer1)
    layer1 = self.pool1(layer1)
    layer1 = self.relu1(layer1)
    layer2 = self.conv2(layer1)
    layer2 = self.conv2_bn(layer2)
    layer2 = self.pool2(layer2)
    layer2 = self.relu2(layer2)
    cout_1 = self.conv3(layer2)
    cout_1 = self.conv3_bn(cout_1)
    cout_2 = self.conv4(cout_1)
    cout_2 = self.conv4_bn(cout_2)
    cout_3 = self.conv5(cout_2)
    cout_3 = self.conv5_bn(cout_3)
    cout_4 = self.conv6(cout_3)
    cout_4 = self.conv6_bn(cout_4)
    pout_1 = self.pool3(cout_4)
    rout_1 = self.relu3(pout_1)
    fc_out = rout_1.view(rout_1.size(0), -1)
    full_1 = self.fc1(fc_out)
    full_1_bn = self.fc1_bn(full_1)
    drop_1 = self.drop1(full_1_bn)
    full_2 = self.fc2(drop_1)
    full_2_bn = self.fc2_bn(full_2)
    out_soft = self.softmax(full_2_bn)
    return out_soft
```

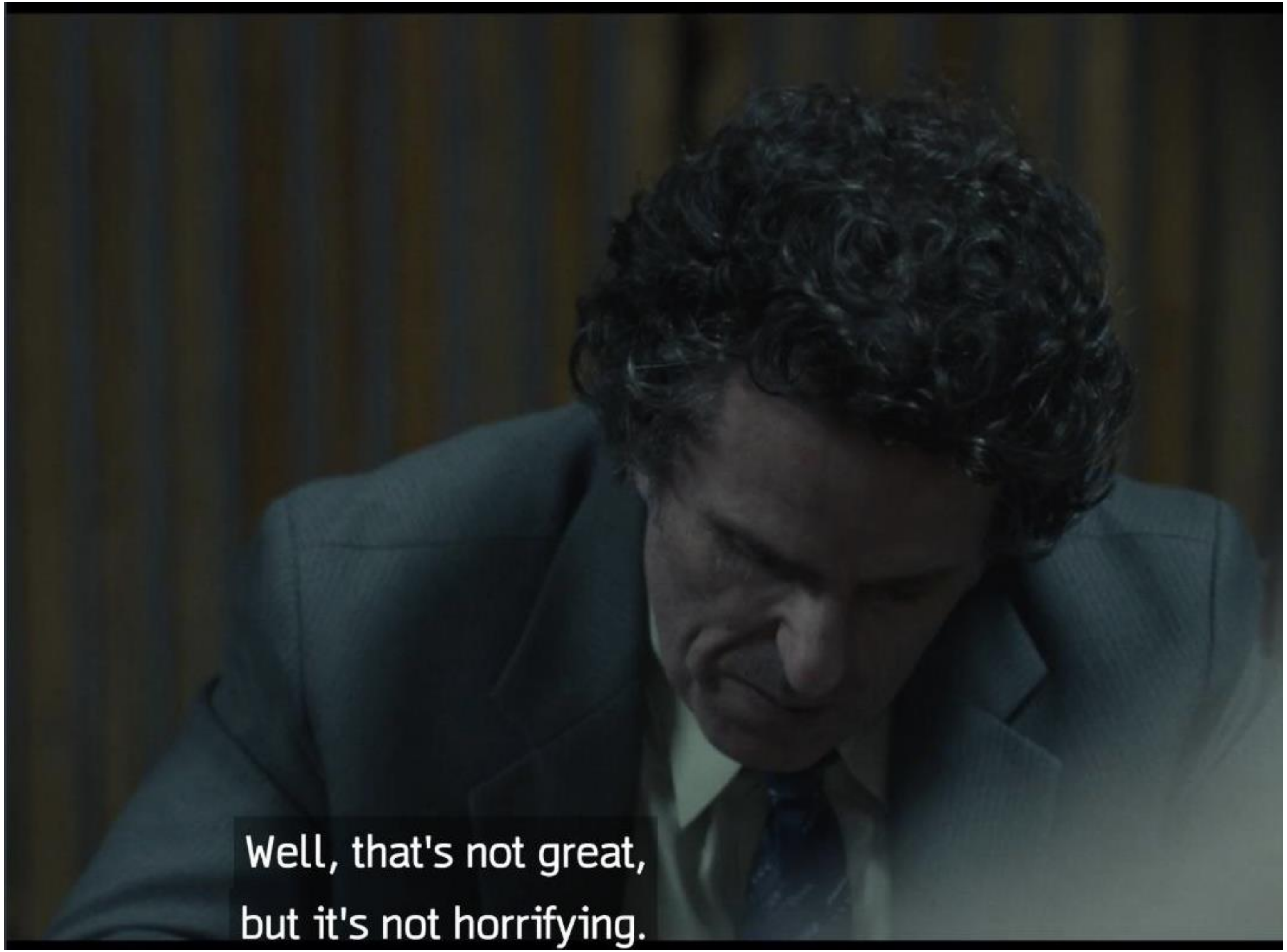
DATA PRE-PROCESSING
ROUND ONE

1. Remove Punctuation
2. Remove PII Flags
3. Strip Newline Characters
4. Lower Case
5. Remove Stopwords
6. Remove Numbers
7. Remove Whitespace

INITIAL FINDINGS

Using the model as specified in the paper:

- F1-score: .71
- Missed the mark on money_transfers
- Ran into lots of OOM issues
- Max sequence length was giving me issues, dropped from 1024 characters to 512
- Clearly more to gain



SOMEWHAT BRIGHT IDEA

- Want:
  - Faster model iterations
  - Ability to use max sequence length 1024

- Solution:
  - Train on a very small subset of the labeled data
  - Test on everything else
  - Randomly sampled 2k from each class

DATA PRE-PROCESSING ROUND TWO

EXPERIMENTS

- Levers to pull
  - Hidden layer size
  - Minibatch size
  - Epochs
  - Learning rate
  - Early stopping

Hyperparamater	**Final Value**
Hidden layer size	256
Batch size	16
Linear size	256
Epochs	500
Early stopping	101
Learning rate	0.005
Training time	~2.5 hours

QUICK ASIDE: TRICKS

CYCLIC LEARNING RATES

- Tuning the learning rate is one of the most important pieces of a neural network
- Approach lets the learning rate vary over epochs, honing in on the best rate for training
- Decreases number of trials that need to be run (saves time and money)

EARLY STOPPING

- Some experiments are just a bad idea
- Stopping them early by computing the loss on a validation set helps to prevent poor results after hours of training
- If the loss on the validation set hasn't improved in N iterations, quit while you're ahead

RESULTS

Class	Precision	Recall	F1-score	Support
bank_service	0.78	0.78	0.78	18,071
credit_card	0.79	0.77	0.78	27,553
credit_reporting	0.78	0.74	0.76	79,234
debt_collection	0.73	0.76	0.75	59,471
loan	0.75	0.74	0.74	29,036
money_transfers	0.94	0.74	0.83	2,734
mortgage	0.86	0.84	0.85	38,281

COMPARISON TO BENCHMARKS

Dataset	Testing Size	Number Classes	Testing Error
Yelp Reviews	50,000	5	40.84%
Yahoo! Answers	60,000	10	29.84%
Amazon Reviews	650,000	5	40.53%
CFPB	254,380	7	20.81%


18476(1)
18437(1)
-92
18594 (1)
18589 (4)
18567 (2)
18514 (1)
18390 (1)
18590(2)
18585(2)
18591(1)
18588(1)
-60
-70
-111

THANK YOU