Multi-label classfication for Reuters text data

Project Report for Deep Learning 2017

Group “Lazy Tigers”
Ari Siitonen, Aaro Salosensaari, Ioanna Bouri
University of Helsinki

December 14, 2017

Contents of the presentation

1. Group work process
2. Dataset handling
3. Notes about models and algorithms (CNNs, LSTMs, embeddings)
4. Final Model and Results

Objective: 15 min presentation

Group work

Brief summary of our process

- Meetings 1-2 per week
- Collaboration on dataset preprocessing methods, metrics, etc.
- + Each member experimented with different models on their own
- Slack: Discussion and sharing Python code

Dataset handling

Reuters newswire dataset

Dataset handling + Training

- ~157k samples, N_CLASSES = 126
- Dataset is very large, what to do about that?
- Zipped XMLs were processed into one large HDF5 file
- HDF5 file was accessed in chunks for training:
  - Chunk of 1000 samples: training / validation split 0.05
  - One chunk of 1000 samples not used for training and retained as a test set.

Text data preprocessing

- Tokenizing
- Word filtering
  - NTLK Stopwords
  - Numbers etc --->
- 2 x headline
- Max 300 words / article
- GloVe Embeddings
  - 131393 / 238172 found
- Word2vec (Google-News)
  - 72441 / 238172 found

```
def tokenize(text, size=300):
    min_length = 3
    text = re.sub(r"[^A-Za-z0-9^,!.\/'+-=]", " ", text)
    text = re.sub(r"what's", "what is ", text)
    text = re.sub(r"\'s", " ", text)
    text = re.sub(r"\'ve", " have ", text)
    text = re.sub(r"can't", "cannot ", text)
    text = re.sub(r"n't", " not ", text)
    text = re.sub(r"i'm", "i am ", text)
    text = re.sub(r"\'re", " are ", text)
    text = re.sub(r"\'d", " would ", text)
    text = re.sub(r"\'ll", " will ", text)
    text = re.sub(r",", " ", text)
    text = re.sub(r"\.", " ", text)
    text = re.sub(r"!", " ! ", text)
    text = re.sub(r"\/", " ", text)
    text = re.sub(r"\^", " ^ ", text)
    text = re.sub(r"\+", " + ", text)
    text = re.sub(r"\-", " - ", text)
    text = re.sub(r"\=", " = ", text)
    text = re.sub(r"'", " ", text)
    text = re.sub(r"(\d+)(k)", r"\g<1>000", text)
    text = re.sub(r":", " : ", text)
    text = re.sub(r" e g ", " eg ", text)
    text = re.sub(r" b g ", " bg ", text)
    text = re.sub(r" u s ", " american ", text)
    text = re.sub(r"\0s", "0", text)
    text = re.sub(r" 9 11 ", "911", text)
    text = re.sub(r"e - mail", "email", text)
    text = re.sub(r"j k", "jk", text)
    text = re.sub(r"\s{2,}", " ", text)
    words = map(lambda word: word.lower(), word_tokenize(text));
    words = [word for word in words
                if word not in cachedStopWords]
    #tokens =(list(map(lambda token: PorterStemmer().stem(token),
    #            words)));
    tokens = words
    filtered_tokens = list(filter(lambda token: re.sub(r"[^A-Za-z0-9^,!.\/'+-=]", '', token),tokens));
    filtered_tokens = list(filter(lambda token: re.sub('[0-9|\-|\.|\,|\/|\+]+', '', token),filtered_tokens))
    filtered_tokens = list(filter(lambda token: re.sub('[?|$|.|!|(|)|:|"|\t|]','',token),filtered_tokens))
    filtered_tokens = list(filter(lambda token: len(token)>=min_length ,filtered_tokens))
    #filtered_tokens = list(filter(lambda text: re.sub(r"n't", " not ", text) and re.sub(r",", '', text) and
    #filtered_tokens = filtered_tokens[:size]
    filtered_tokens = list(filter(None and '-', filtered_tokens))

    return filtered_tokens
```

Models and algorithms

Embeddings, CNNs, LSTMs

Activation & Loss function

- In all our models:
- Final layer: Dense layer with N_CLASSES = 126 outputs
- Sigmoid activation (each label normalized -> [0,1])
  - Probability “is this label present in this data sample”
  - Softmax would assign a probability to each class, but this is multi-label classification
- Binary crossentropy
  - Mean of per-label binary crossentropy

Network models we tried

- With & without embedding
  - (Glove and Word2vec Gensim)
- MLP, CNN, LSTM
- CNN with varying kernel window size
- LSTM with uni/bidirectional layers
- CNN + LSTM combo
- With/without class weights
- Best models: Relatively simple CNNs with embeddings
- The final model: CNN model with Gensim embedding

Optimization

- Optimizers' ranking:
  - RMSProp
  - Nadam
  - Adagrad
  - Adam
  - Adadelta
  - SGD
  - Adamax
- All optimizers had a very slight difference in accuracy
- RMSprop usually seemed to perform the best

- Various batch sizes and epoch lengths
- Final setup:
  - Batch size 256 samples
  - 4 epochs per chunk
  - 150 chunks = 600 epochs

Results

- The Final Model
- + Brief look at intermediate models

The Final Model We Submitted

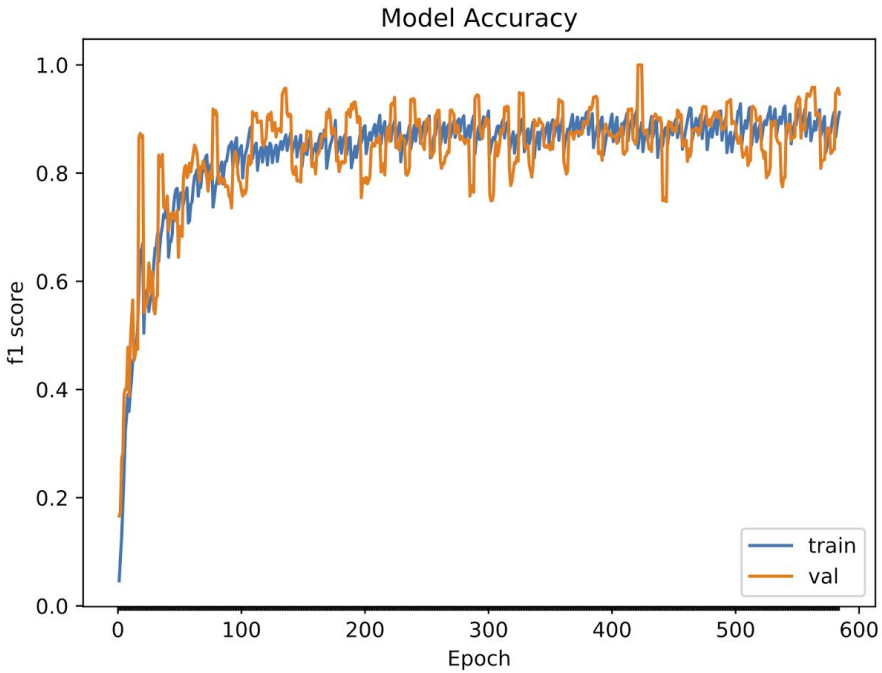
CNN+GlobalMaxPooling + Gensim

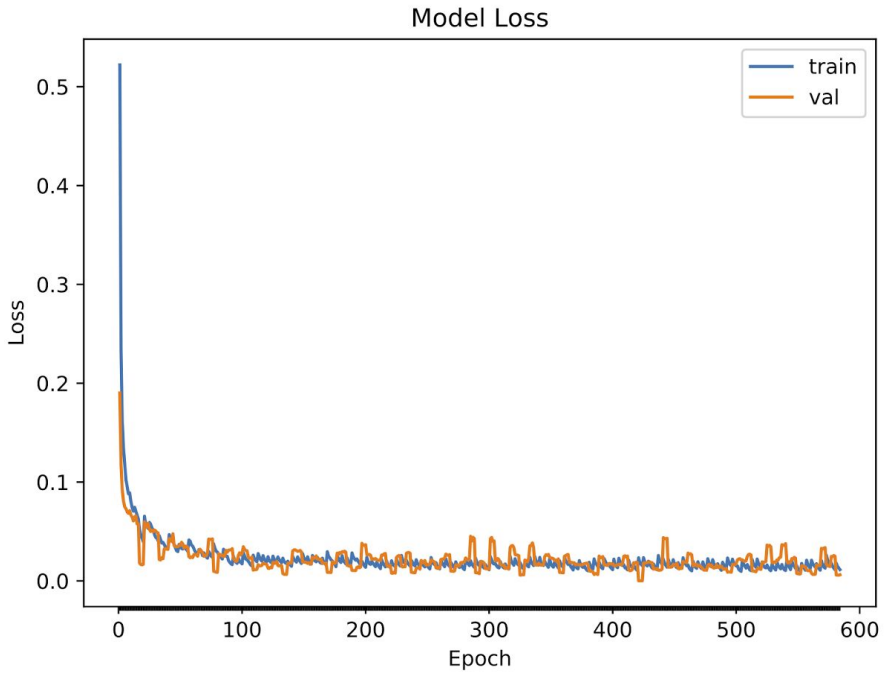
```
batch_size=256
epochs = 4
x_shape = 300
y_shape = 126

N_CLASSES = 126


sequence_input = Input(shape=(x_shape,), dtype='int32')
embedded_sequences = embedding_layer2(sequence_input)
x = Dropout(0.3)(embedded_sequences )
x = Conv1D(300, 10, activation='relu')(x)
x = MaxPooling1D(5)(x)
x = Conv1D(100, 8, activation='relu')(x)
x = GlobalMaxPooling1D()(x)
x = Dropout(0.2)(x)
x = Dense(N_CLASSES)(x)


ma_cnn4_preds = Activation('sigmoid')(x)

model = Model(inputs=sequence_input, outputs=ma_cnn4_preds)

print(model.summary())
model.compile(optimizer='RMSprop',
                loss='binary_crossentropy',
                metrics=[f1_score])
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_5 (InputLayer)         (None, 300)               0
_________________________________________________________________
embedding_3 (Embedding)      (None, 300, 300)          71451900
_________________________________________________________________
dropout_7 (Dropout)          (None, 300, 300)          0
_________________________________________________________________
conv1d_7 (Conv1D)            (None, 291, 300)          900300
_________________________________________________________________
max_pooling1d_5 (MaxPooling1 (None, 58, 300)           0
_________________________________________________________________
conv1d_8 (Conv1D)            (None, 51, 100)           240100
_________________________________________________________________
global_max_pooling1d_3 (Glob (None, 100)               0
_________________________________________________________________
dropout_8 (Dropout)          (None, 100)               0
_________________________________________________________________
dense_5 (Dense)              (None, 126)               12726
_________________________________________________________________
activation_3 (Activation)    (None, 126)               0
=================================================================
Total params: 72,605,026
Trainable params: 1,153,126
Non-trainable params: 71,451,900
_________________________________________________________________
None
```

Test set metrics:

```
test set metrics:
exact match accuracy
0.695
precision (true pos scores / all pred pos)
0.932559010865
recall (true pos scores / all real pos)
0.829666666667
f1
0.878109013935
jaccard
0.851763492063
hamming loss
0.0054841269841269845
Confusion matrix:
[[122820    180]
 [   511   2489]]
```





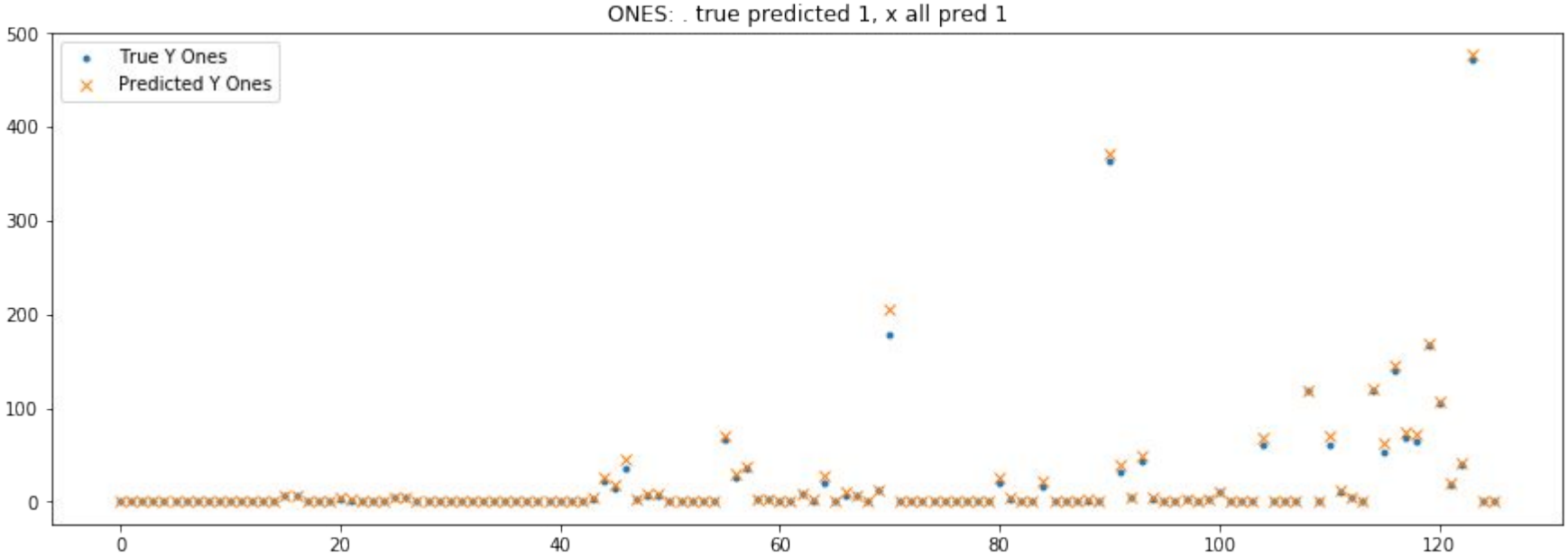
F1 score + Loss function during training (train set + 0.05 split validation set)

Test set metrics for some intermediate models

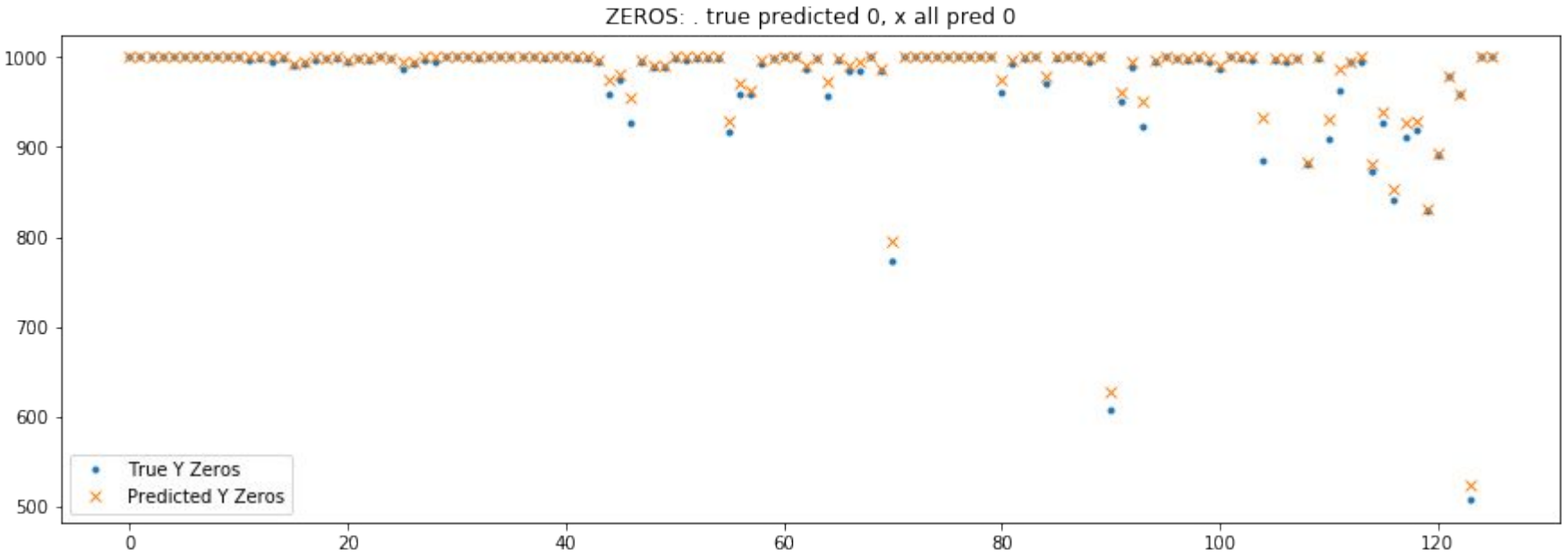
- 2-layer (CNN + MaxPooling1D), with GloVe
  - F1 0.8664, precision 0.8905, recall 0.8436
- Same as ^ + Inverse class weights
  - F1 0.8062, precision 0.8480, recall 0.7683
- CNN + MaxPooling1D + CNN + GlobalMaxPool, with GloVe
  - F1 0.8671, precision 0.9365, recall 0.8073
- Same as ^ but with word2vec (= Final Model)
  - F1 0.8781, precision 0.9325, recall 0.8296
  - 0.695 exact match accuracy!

If we still have time left -> more plots

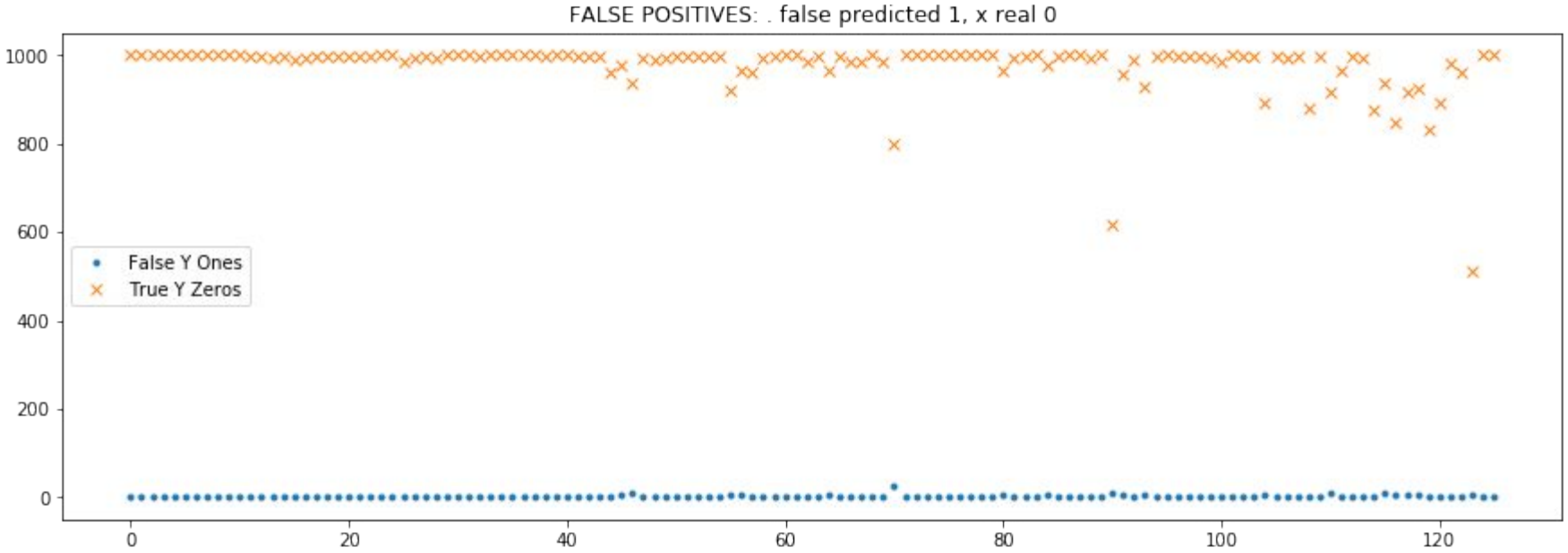
Prediction performance of the Final Model



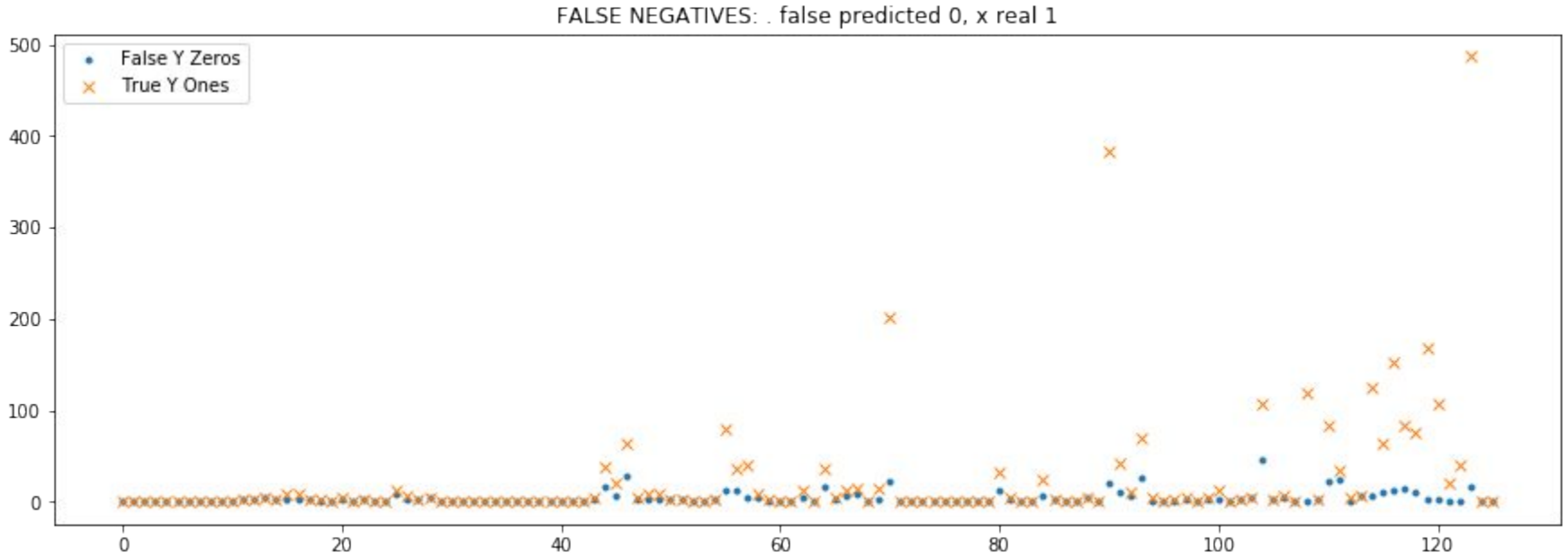
How many of all predicted ones (x) were predicted correctly (o)?



How many of all predicted zeros (x) were predicted correctly (o)?



How many of the zeros in test set (x) were mis-labeled as ones (o)?



How many of the ones in test set (x) were mis-labeled as zeros (o)?

Thank you!

Appendix: Additional model definitions + plots

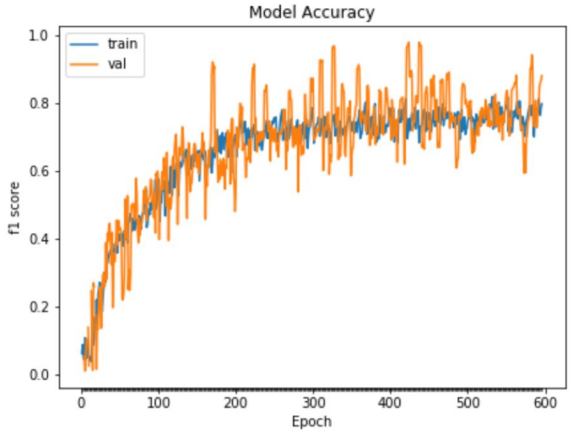
—

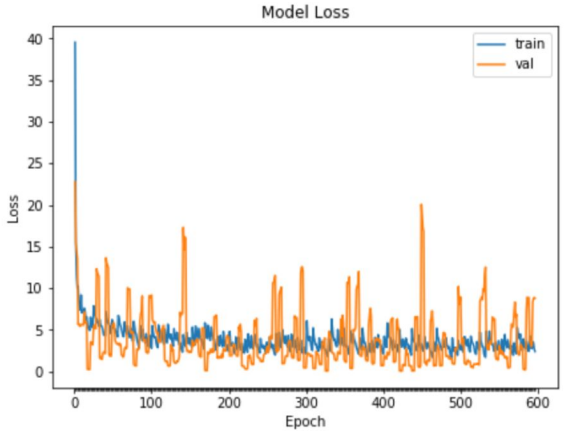
Note:

- We could not fit all of our experiments in a notebook / pdf with max length 20 pages
- So they are included here for reference
- NOT covered in the presentation
- TODO: Screenshots of everything that wasn't in the notebook we submitted

CNN+Inverse weights

```
f1_score 0.815966430664
(1000, 126)
test set metrics:
exact match accuracy
0.534
precision (true pos scores / all pred pos)
0.848050036792
recall (true pos scores / all real pos)
0.768333333333
f1
0.80622595313
jaccard
0.754646608947
hamming loss
0.008793650793650794
Confusion matrix:
[[122587    413]
 [   695   2305]]
(1000, 126)
```





The model definition for CNN

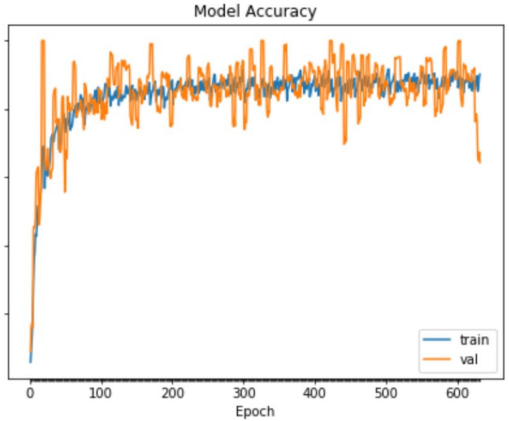
```
sequence_input = Input(shape=(x_shape,), dtype='int32')
embedded_sequences = embedding_layer(sequence_input)
x = Dropout(0.3)(embedded_sequences)
x = Conv1D(300, 8, activation='relu')(x)
x = MaxPooling1D(5)(x)
x = Conv1D(100, 5, activation='relu')(x)
x = MaxPooling1D(5)(x)
x = Flatten()(x)
x = Dropout(0.2)(x)
x = Dense(128, activation='relu')(x)
preds = Dense(y_shape, activation='sigmoid')(x)

model = Model(sequence_input, preds)
model.summary()
model.compile(loss='binary_crossentropy',
              optimizer='RMSprop',
              metrics=[f1_score])
```

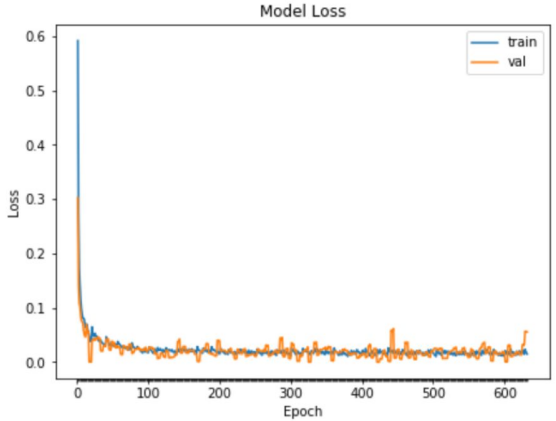
During training:

```
class_sums1 = y_data.sum(axis=0)
class_sums1 = np.where(class_sums1==0,0.001, class_sums1)
inverse_frequencies = (y_data.sum()/class_sums1)
class_weight = dict((i, round(inverse_freq)) for i, inverse_freq in enumerate(inverse_frequencies))
```

CNN + MaxPool1d

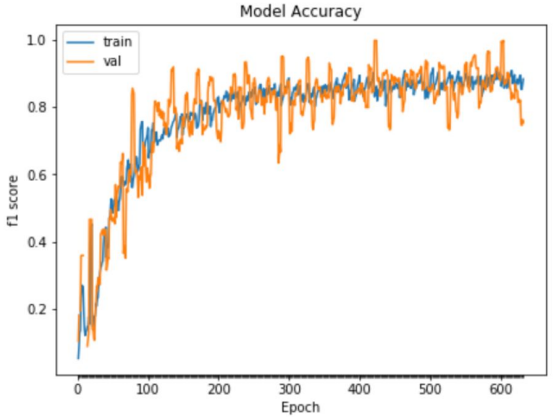
```
f1_score 0.873879217148
(1000, 126)
test set metrics:
exact match accuracy
0.679
precision (true pos scores / all pred pos)
0.936581593194
recall (true pos scores / all real pos)
0.807333333333
f1
0.867167919799
jaccard
0.838249603175
hamming loss
0.005888888888888889
Confusion matrix:
[[122836    164]
 [   578   2422]]
```

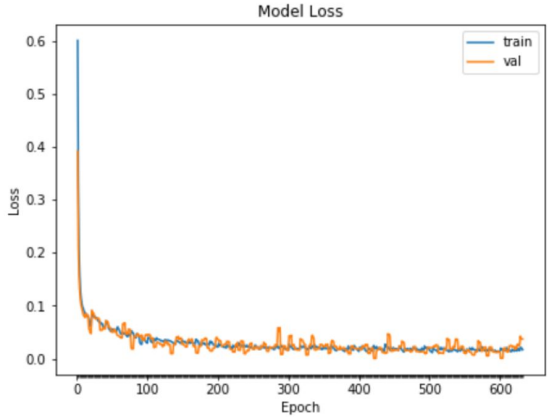




(Model definition same as previous slide, but without the class weights)

CNN+LSTM



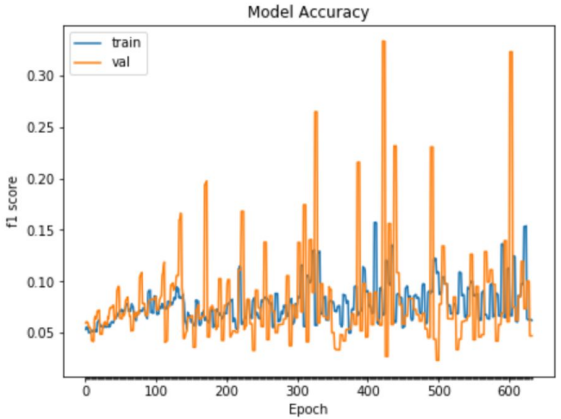


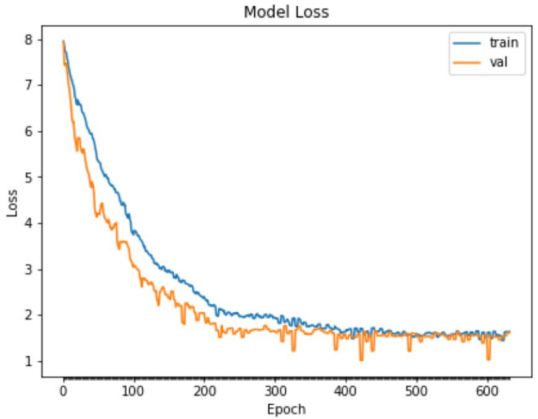
	precision	recall	f1-score	support
zero	1.00	1.00	1.00	123000
one?	0.93	0.81	0.87	3000
avg / total	0.99	0.99	0.99	126000

CNN + LSTM model definition

```
sequence_input = Input(shape=(x_shape,), dtype='int32')
#embedded_sequences = Embedding(len(word_index) + 1, EMBEDDING_DIM, input_length=MAX_SEQUENCE_LENGTH)(sequence_input)
embedded_sequences = embedding_layer(sequence_input)
x = Dropout(0.2)(embedded_sequences)
x = Conv1D(100, 5, activation='relu')(x)
x = MaxPooling1D(5)(x)
x = LSTM(100)(x)
x = Dropout(0.2)(x)
preds = Dense(y_shape, activation='sigmoid')(x)

model = Model(sequence_input, preds)
model.summary()
model.compile(loss='binary_crossentropy',
              optimizer='RMSprop',
              metrics=[f1_score])
```

MLP





	precision	recall	f1-score	support
zero	0.98	0.92	0.95	123000
one?	0.06	0.21	0.10	3000
avg / total	0.96	0.91	0.93	126000