CS 224U

Bake-off 2: Sentiment Analysis

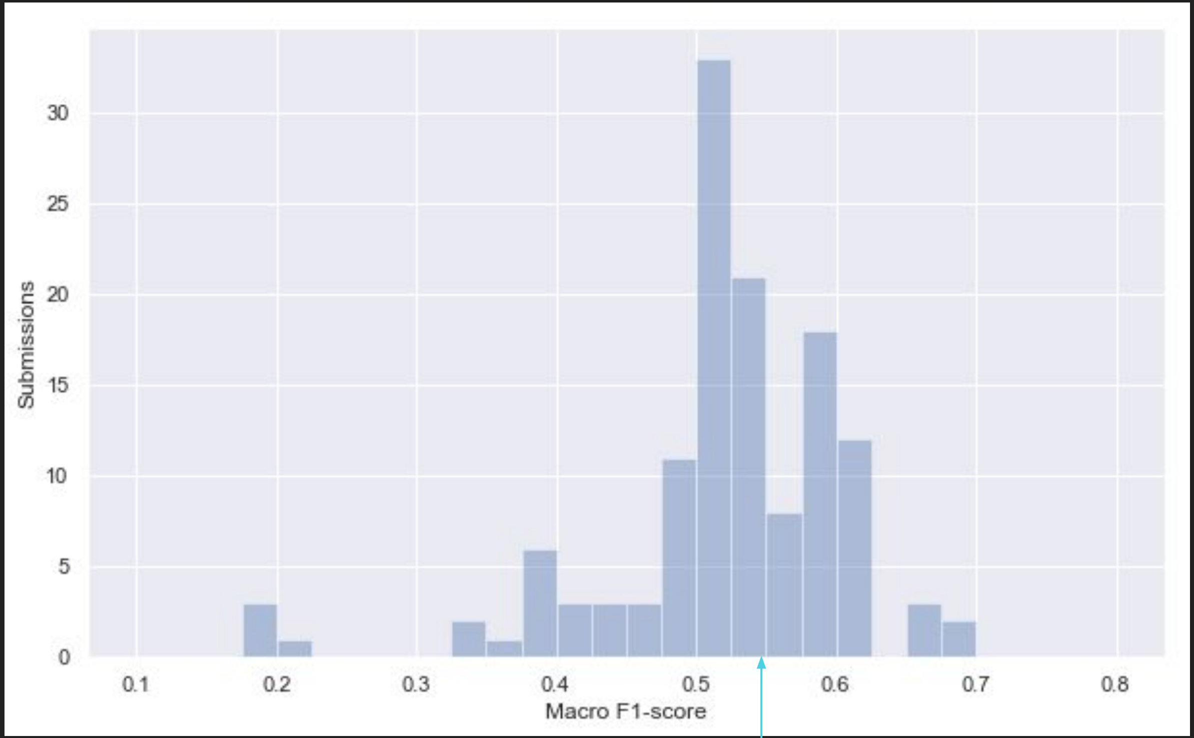
:)   :| :(

Cindy & Jayadev

Task

- Sentiment analysis with 3 classes: positive, neutral, negative
- Evaluation: Stanford Sentiment Treebank Test Set
  - 2210 sentences in test set
- Evaluation metric: Macro F1 score (**NOT** micro F1 or weighted macro F1)
  - ~900 positive, ~400 neutral, ~900 negative
  - In general, worst performance seen on “neutral” class

Histogram of scores



unigrams_phi + softmax

What distinguishes the high scorers?

High o/e for top scorers (>= 0.58)

	top	bottom
dev	2.061939	0.326789
y_dev	2.034808	0.343988
f	2.004663	0.363099
sst_train	1.996595	0.368213
sst_dev	1.979760	0.378886
bert_sentence_phi	1.963751	0.389035
y_train	1.958842	0.392147
torch.long	1.952593	0.396108
hidden_size	1.946218	0.400150
t.leaves	1.923450	0.414583
X_bert_train_mean	1.921352	0.415913
train	1.914657	0.420157
/	1.911065	0.422435
X_str_train	1.910325	0.422903
X_bert_train	1.907963	0.424401
batch	1.905052	0.426247
X.mean	1.905052	0.426247
BERT	1.899154	0.429986
context	1.899154	0.429986
X_bert_dev	1.892795	0.434017

High o/e for low scorers (<0.58)

	top	bottom
np_func	0.181509	1.518879
score	0.195754	1.509848
rnn_phi	0.204557	1.504267
feats	0.213304	1.498722
glove_subtree_phi	0.226090	1.490617
lookup	0.229404	1.488516
np.sum	0.230127	1.488057
sst_glove_vocab	0.289769	1.450247
0.05	0.303226	1.441716
DATE_HOME	0.305796	1.440087
0.001	0.306836	1.439428
sst_train_vocab	0.310245	1.437267
get_vocab	0.315603	1.433870
avg	0.318200	1.432224
vector	0.332571	1.423114
iter	0.334110	1.422138
10000	0.336186	1.420822
words	0.350329	1.411856
6B	0.351467	1.411135
negative	0.358882	1.406434

What distinguishes the high scorers?

High o/e for top scorers (>= 0.58)

	top	bottom
dev	2.061939	0.326789
y_dev	2.034808	0.343988
f	2.004663	0.363099
sst_train	1.996595	0.368213
sst_dev	1.979760	0.378886
bert_sentence_phi	1.963751	0.389035
y_train	1.958842	0.392147
torch.long	1.952593	0.396108
hidden_size	1.946218	0.400150
t.leaves	1.923450	0.414583
X_bert_train_mean	1.921352	0.415913
train	1.914657	0.420157
/	1.911065	0.422435
X_str_train	1.910325	0.422903
X_bert_train	1.907963	0.424401
batch	1.905052	0.426247
X.mean	1.905052	0.426247
BERT	1.899154	0.429986
context	1.899154	0.429986
X_bert_dev	1.892795	0.434017

High o/e for low scorers (<0.58)

	top	bottom
np_func	0.181509	1.518879
score	0.195754	1.509848
rnn_phi	0.204557	1.504267
feats	0.213304	1.498722
glove_subtree_phi	0.226090	1.490617
lookup	0.229404	1.488516
np.sum	0.230127	1.488057
sst_glove_vocab	0.289769	1.450247
0.05	0.303226	1.441716
DATE_HOME	0.305796	1.440087
0.001	0.306836	1.439428
sst_train_vocab	0.310245	1.437267
get_vocab	0.315603	1.433870
avg	0.318200	1.432224
vector	0.332571	1.423114
iter	0.334110	1.422138
10000	0.336186	1.420822
words	0.350329	1.411856
6B	0.351467	1.411135
negative	0.358882	1.406434

Using BERT for feature extraction and fine-tuning seems to be very effective.

🥇 Group 13 (Di B., Yipeng H., Zijian W.) Score: 0.692

Balanced Dataset + End-to-end BERT

- Data preprocessing:
  - Balance the dataset by oversampling
  - Filter sentences to rejoin contractions and punctuation:

```
def sent_filter(sent):
    return sent.replace(" 's", "'s").replace(" .", ".").replace(" ,", ",").replace("`` ", "''") \
            .replace(" ''", "''").replace(" 'm", "'m").replace(" 've", "'ve") \
            .replace(" 't", "'t").replace(" 're", "'re")
```

- End-to-end BERT:
  - Train the model using the [pretrained BERT model in PyTorch]
  - Use hyperparameter settings from original BERT paper

🥈 Group 51 (Hanoz B., Angelia R. W.)
Score: 0.651

BERT + TorchShallowNeuralClassifier + Balanced Dataset

- BERT encoder:
  - Fine-tune BERT on the SST
  - Run inference to generate features for each sentence
- Classifier:
  - Use TorchShallowNeuralClassifier
  - Up-sample the instances with class 'neutral' during training to ensure roughly balanced dataset

Other interesting approaches

Group 9
Score: 0.69 using the subtree labels (disallowed in the competition but interesting in general)

Seq2seq

- Intuition:
  - Strings containing sentence annotations and tree structure as input sequence
  - Sentiment label as output “sequence”
- Architecture:
  - 2-layer bidirectional LSTM encoder/decoder with multiplicative attention

Other interesting approaches

Feature engineering

- All top systems this year relied on deep learning
- Last year's top 2 systems both used hand-built features + logistic regression
    - Note: scores below are on the binary task
    - First place (Jayadev's team!)
        - Score: 0.831
        - Preprocessing: Remove punctuation
        - Features: Character n-grams, tf-idf weighting
        - Classification: Logistic regression with balanced class weight
    - Second place (Lucy's team!)
        - Score: 0.821
        - Preprocessing: Remove stopwords
        - Features: Unigrams/bigrams, negation words, sentiment lexicon, part of speech, sentence length, GloVe