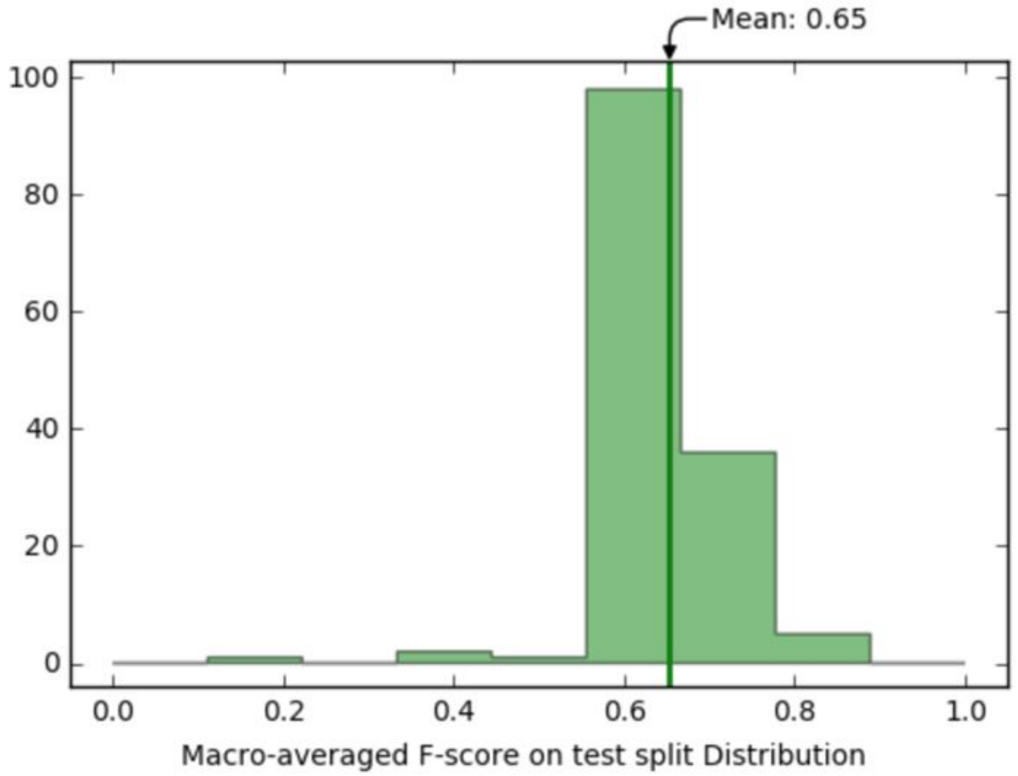
CS224U Bake-off 3 Discussion

Spr-1718 Allen Nie

Overall Distribution



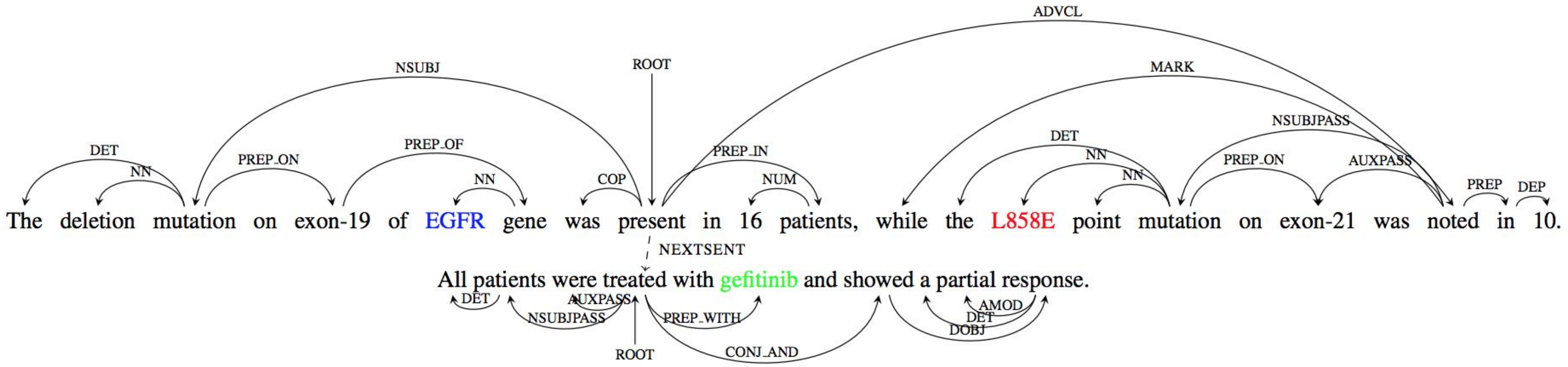
Preprocessing and feature selection

1. Many people decide to use Porter Stemmer
2. Various bag of words n-gram techniques are used (unigram, bigram, trigram)
3. Many explored L1 vs. L2 penalty (seems to help some but not others)
4. Number of tokens in left/right/middle, POS tags, bag-of-POS tags.
5. A few groups played with GloVE and Word2Vec embeddings and got great results
6. Character-level model does not seem to shine as much this time (24th, 25th, 39th, 78th)
7. Random Forest classifier seems to handle high-dimension features quite well!
8. Dependency parsing could have helped, but no one used it!

Preprocessing and feature selection

Besides left/right/middle, could derive dependency path-based features :)

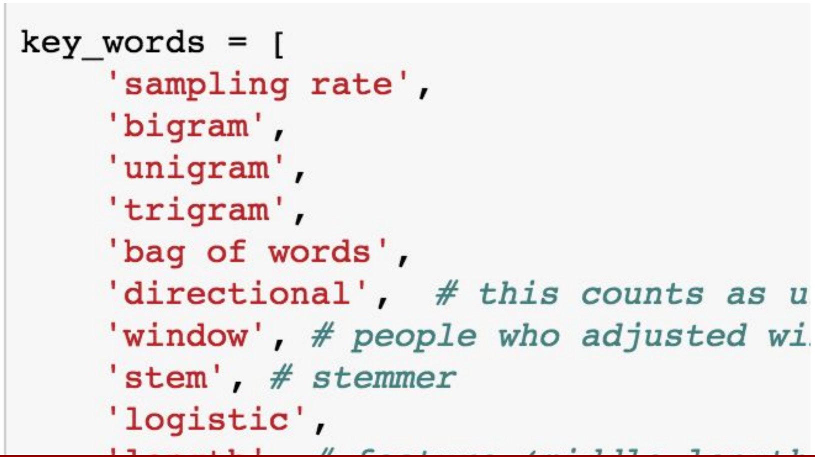
Peng, Nanyun, et al. "Cross-sentence n-ary relation extraction with graph lstms." *arXiv preprint arXiv:1708.03743* (2017).


The deletion mutation on exon-19 of EGFR gene was present in 16 patients, while the L858E point mutation on exon-21 was noted in 10.
All patients were treated with gefitinib and showed a partial response.
ADVCL
NSUBJ
ROOT
MARK
NSUBJPASS
DET
NN
PREP_ON
PREP_OF
COP
PREP_IN
NUM
AUXPASS
PREP
DEP
NEXTSENT
AMOD
DOBJ
PREP_WITH
CONJ_AND

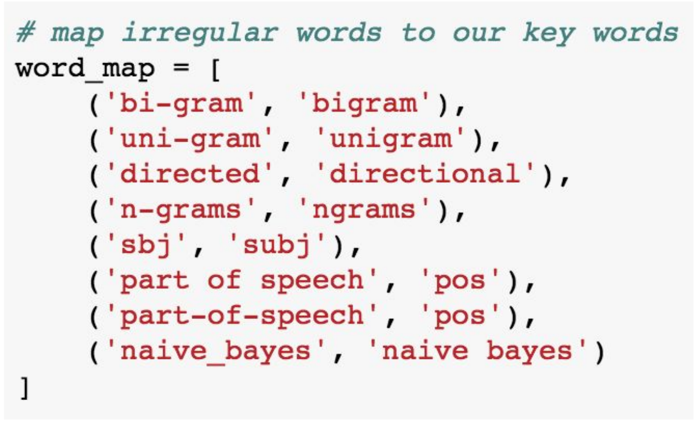
Knowledge Mining from Language

Since we know the common words describing features and classifiers, can we use what we learned from this class to write a featurizer?

Simple! You’ve all done this a couple of dozen times. Let’s just write a simple bag of words featurizer…



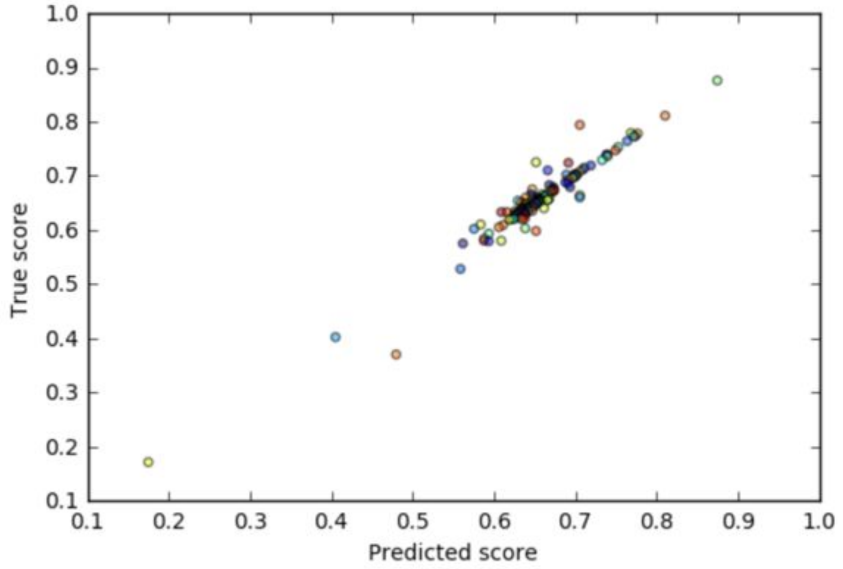
```
key_words = [
    'sampling rate',
    'bigram',
    'unigram',
    'trigram',
    'bag of words',
    'directional',  # this counts as u
    'window', # people who adjusted wi
    'stem', # stemmer
    'logistic',
```



```
# map irregular words to our key words
word_map = [
    ('bi-gram', 'bigram'),
    ('uni-gram', 'unigram'),
    ('directed', 'directional'),
    ('n-grams', 'ngrams'),
    ('sbj', 'subj'),
    ('part of speech', 'pos'),
    ('part-of-speech', 'pos'),
    ('naive_bayes', 'naive bayes')
]
```

Building a Good Prediction Model

We allow our model to be non-linear, and allow interaction between these features. Here is how well our model does!



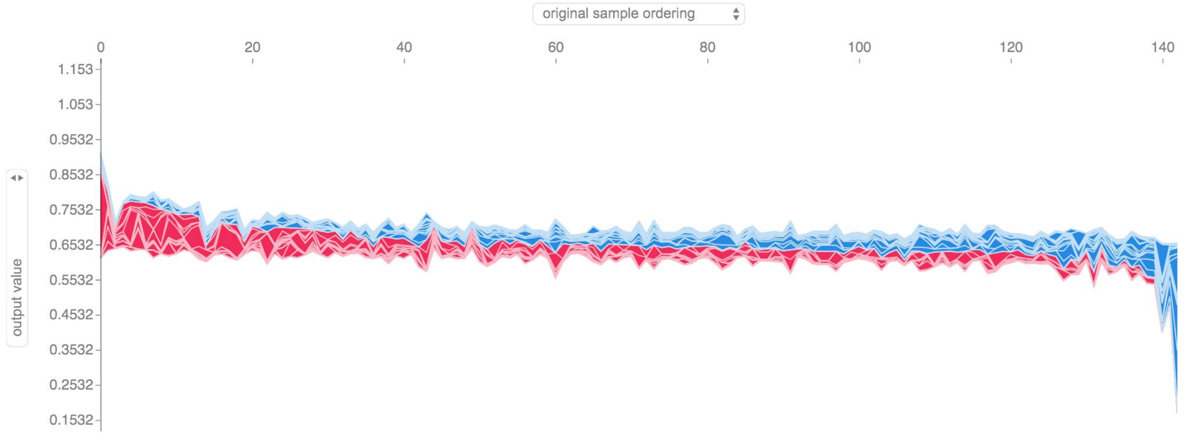
Prediction vs. True F-0.5 Score Plot

MSE = 0.0003094

Prediction is Never Enough!

The features more associated with top-performing models will contribute positively to the predicted score, and features associated with low-performing model will contribute negative to the predicted score.

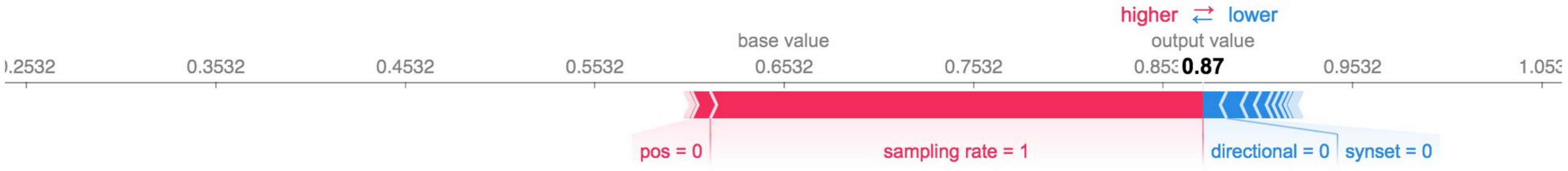
But keep in mind that bag of words model takes no **syntax** into account. More complex models like LSTM will, but it's also hard to interpret.



Top 5

F1 Score: 0.876 (jessepb)

I lowered the sampling rate to 0.001 and implemented a simple bigrams featurizer.

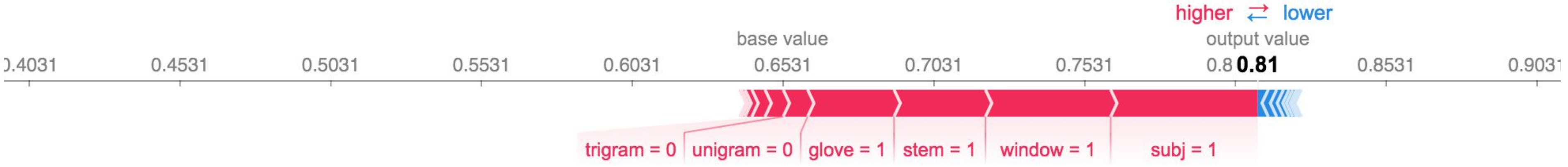


Top 5

F1 Score: 0.811 (jmendels, lucy3)

We used several featurizers in our model:

- We modified the directional bag of words featurizer from the homework to include one word to the left of the first entity and one word to the right of the last entity. We found that a bigger window to the left or right decreased performance. We also used a NLTK porter stemmer in this function.
- We also had a bigram version of our directional bag of words featurizer, which also included the single-token window to the left and right and a porter stemmer.
- We included three average lengths as features: the length of all middles, the length of subject->object middles, and the length of object->subject middles.
- We included synset_featurizer and middle_bigram_pos_tag_featurizer from Homework 3.
- We took the last token in each entity and appended their GloVe vectors as additional featuresin glove.6B.50d.txt, as well as the cosine similarity between them.
- In the end, We have a total of 231,276 features and had a score of 0.813 on the dev set.



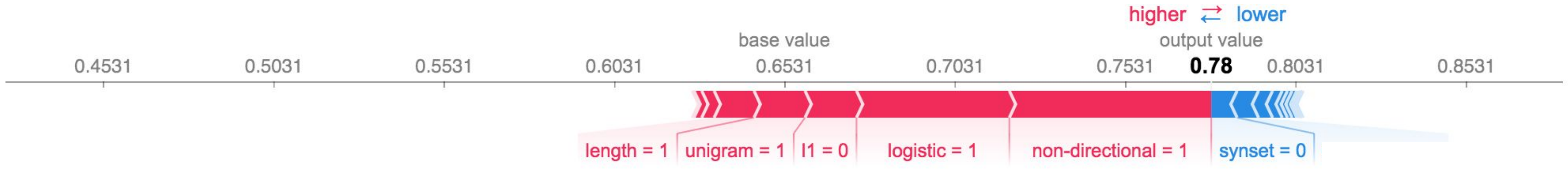
Top 5

F1 Score: 0.794 (surag)

directed unigrams for middle, and 3 words each to the left and right

F1 Score: 0.779 (nmakow)

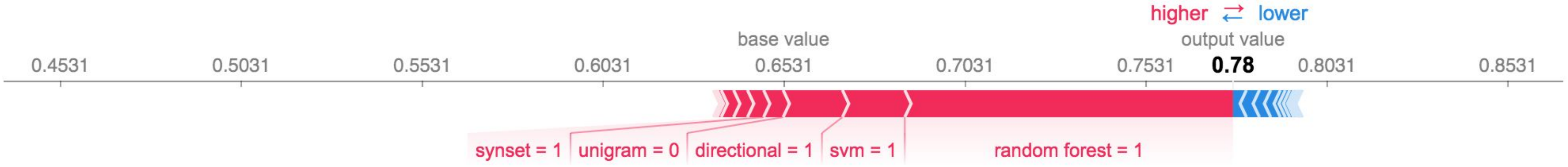
Baseline Logistic Regression model with following features:
- Directional unigrams and bigrams of middle
- # of tokens in middle, left, and right contexts
- Directional POS tag of middle words (bigrams)
- (Non-directional) unigrams and bigrams of left, right contexts



Top 5

F1 Score: 0.778 (greis, minhan)

We use the features from all the following featurizers: directional_bag_of_words_featurizer, middle_bigram_pos_tag_featurizer, synset_featurizer, middle_bigram_tag_featurizer, mention_pos_sub_obj_featurizer. The last two are our own -- middle_bigram_tag_featurizer simply gets the bigrams for the middle part, and the last one appends obj_ or sub_ to the mention words, respectively. Doing this we ended up with over 100 thousand dimensions, so we ruled out trying to make a distance-based method (eg SVMs work) as they would mostly pick up noise; instead we needed something robust to high-dimensionality so we tried a Random Forest. With unlimited depth and 100 trees, we already got great results, but we increased to 200 trees for a slightly better F-score of 0.778. Increasing trees to 1k helps a bit, but not more than a couple percent.



Other interesting features and models

(etang21, zmarilyn)

Using the 50 dimensional GloVe dictionary from the VSM unit, we added a feature for each component of the subject's vector representation and a feature for each component of the object's vector representation. Adding this featurizer to the homework baseline brought us to 0.725 on the dev set.

We used a set of 300-dim Word2vec vectors which contained feature representations for multi-word entities (also, a ridiculous 3 million word vocab). About half the KB entities appear in this VSM. Using the gensim package, we then repeated the same process as earlier to featurize. On the test set, this Word2vec featurizer, our GloVe featurizer, and the HW featurizers reached 0.747 .

(patricko)

I figured that the matrices in train_model were enormous and pretty sparse, so I thought why not try and reduce their dimension? So, I used sklearn.decomposition's TruncatedSVD on each matrix in train_model and predict. However, it was far too resource-intensive to reduce down to any more than a couple hundred dimensions, so I think the dimensionality reduction was far too 'reducing', which explains why the scores were so low. I used all the synset featurizer from homework 3.

Thank you!