CIS 530: Homework 1

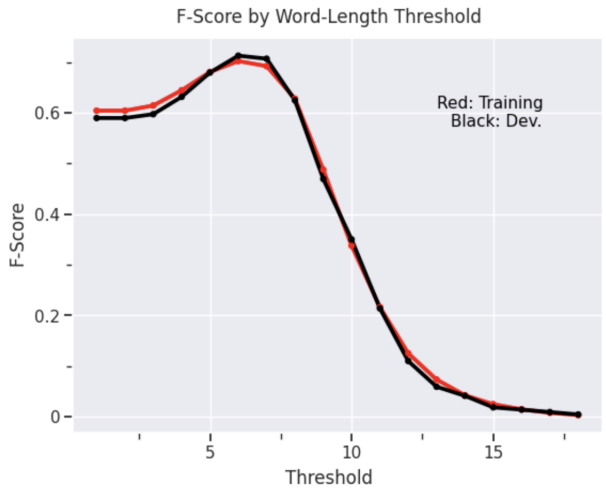
Octavio E. Lima
Alexander Dong

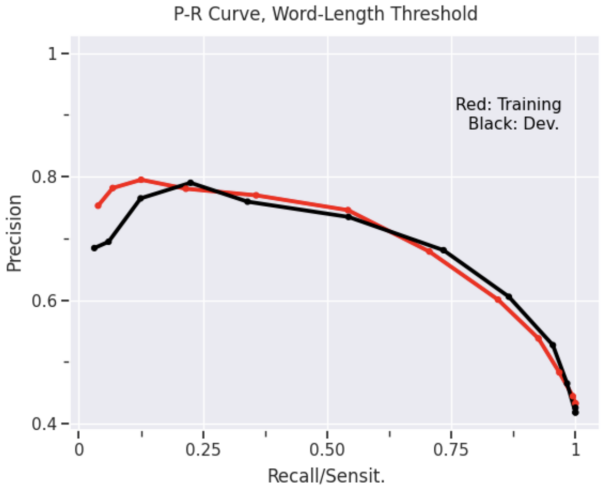
September 19, 2022

1. Baselines

	Train			Dev		
Classifier	Precision	Recall	F-Score	Precision	Recall	F-Score
All-Complex	0.468	1.000	0.638	0.454	1.000	0.624
Word-Length Baseline	0.645	0.838	0.729	0.652	0.861	0.742
Word-Freq Baseline	0.614	0.809	0.698	0.600	0.838	0.699
Naive Bayes	0.495	0.980	0.658	0.469	0.969	0.632
Logistic Regression	0.716	0.643	0.677	0.723	0.687	0.704

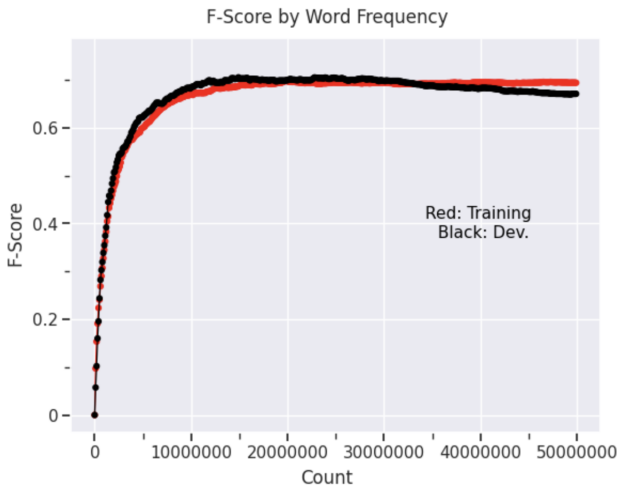
(a) Results from Word-Length Model:

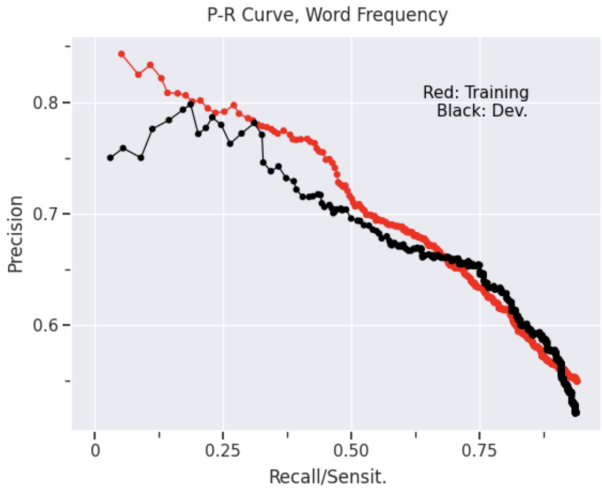




As we can observe, the optimum threshold for the word-length-based model is 6 characters, where F-Score reaches the value of 0.728884 in the training dataset, and 0.741693 in the development dataset.

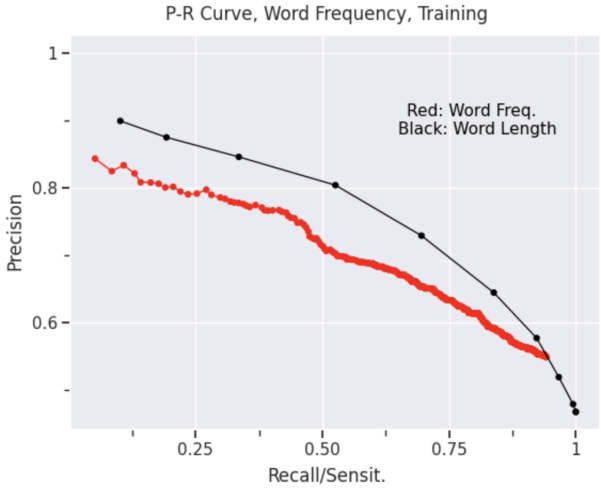
(b) Results from Word-Frequency Model:

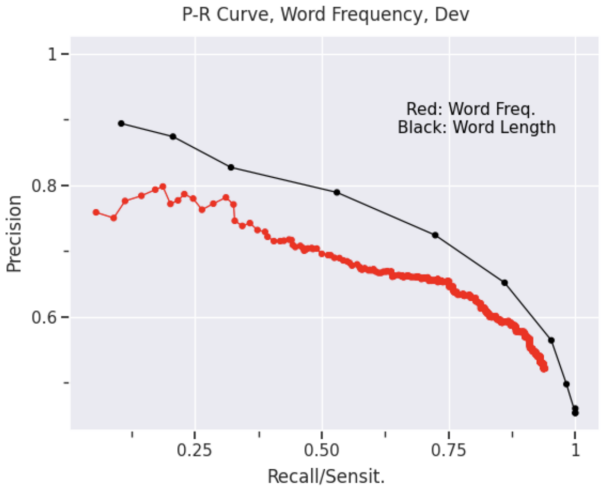




Based on the figures above (and our own findings), the threshold that maximizes F-Score on the training data is 19,900,000. For the Development dataset, that number is 14,900,000.

(c) Comparison:





In summary, the Word-Length-Based Model outperforms the Word-Frequency Model. It has substantially higher precision for almost all values of Recall in the x-axes.

2. Classifiers

(a) Naive Bayes

i. Our Naive Bayes classifier has performed better than the "All Complex" model. Nonetheless, it has failed to outperform the other Simple Baselines.

ii. Its F-Scores were respectively 0.658 and 0.632 on the Training and Development datasets.

Classifier	F-Score on Train	F-Score on Dev
All-Complex	0.638	0.624
Word-Length Baseline	0.729	0.742
Word-Freq Baseline	0.698	0.699
Naive Bayes	0.658	0.632
Logistic Regression	0.677	0.704

(b) Logistic Regression

i. The table above indicates that our Logistic Regression has reached a higher F-Score than Naive Bayes for both the Training *and* Development datasets.

ii. Respectively, its Training and Development F-Scores were 0.677 and 0.704 — with the difference being higher in the Development Dataset.

(c) Logistic Regression v. Naive Bayes

i. According to DataEspresso (2017), Logistic Regression is a **linear classification method**, whereas Naive Bayes is "a **classification method based on Bayes' theorem** that derives the probability of the given feature vector being associated with a label."

ii. Essentially, the two models work in similar ways. The main difference is the assumptions that they consider prior to the analysis.

iii. "Naïve Bayes has a naive assumption of conditional independence for every feature, which means that the algorithm expects the features to be independent which not always is the case." On the other hand, in the case of Logistic Regressions, the features are divided linearly.

iv. In our case, this helps explain the difference in F-Score performance between the two models — and, specifically, the noticeable difference in F-Score performance in the Development Dataset.

3. Adaboost

The performance of the adaboost model we have implemented is depicted below.

	Train			Dev		
Classifier	P	R	F	P	R	F
Adaboost	0.746	0.769	0.758	0.721	0.763	0.741

(a) We used an adaboost model with a decision tree classifier as the base classifier. We chose this model because we knew ensemble techniques work well in classification problems. To find the best hyperparameters, we used a grid search on the parameters max depth, min sample leaves, number of estimators, and learning rate. We tried using a random forest model but the random forest model over fit on the data.

the results for the training set were:

(P: .863, R: .801, F: .832)

while the results for the validation set were:

(P: .693, R: .689, F: .683)

Although the adaboost model had worse training set results, the validation set were much better at (P: .720, R: .763, F: .741).

Thus, we can see that Adaboost is better at generalizing from the training set onto the development set, therefore most likely better at generalizing onto the test set.

Classifier	F-Score on Train	F-Score on Dev
All-Complex	0.638	0.624
Word-Length Baseline	0.729	0.742
Word-Freq Baseline	0.698	0.699
Naive Bayes	0.658	0.632
Logistic Regression	0.677	0.704
Adaboost	**0.758**	**0.741**

(b) For the features, we used number of syllables and number of word net synonyms. If a word is complex, then the number of syllables is probably high. That is why we think syllables is a good feature to train on. If a word is complex, then there will be very few words that have the same meaning.

(c) Some words that we predicted as complex (43 percent of words) but weren't are the following:

('continues', 'overthrown', 'personal', 'firefighters', 'praising', 'elementary', 'interim', 'businesses', 'minister', 'laughter', 'threatened', 'condition', 'cheeseburgers', 'long-term', 'briefing', 'manipulating', 'handprints', 'spontaneous', 'fireworks', 'prisons')

On the other hand, the words that we couldn't predict as complex (57 percent of words) were:

('coup', 'lilting', 'toughest', 'evaded', 'recruit', 'kinky', 'secure', 'concept', 'scuffle', 'jolted', 'steamy', 'weakness', 'pitted', 'retained', 'worded', 'research', 'plagued', 'subdued', 'grooms', 'material')

(d) In general, the average length of false positives (we thought they were complex) is 9 characters, while the average length of false negatives is 6. This shows that our model does poorly against long non-complex words and short complex words. When looking at the number of syllables, we get a similar story. We believe that the number of syllables is closely connected to the length of the word, thus we have 2 features that are related to the length of the word. This means that our model probably weighs the length very highly and thus will fail on short complex words and long non-complex words.