Classification of Complex Words
CSC306- NLP
Project 2 Report

Kurtik Appadoo

1 Background

In text classification tasks, particularly for complex word identification, several key evaluation metrics are essential to assess model performance.

- Accuracy: The ratio of correctly predicted labels to total predictions
- Precision: The ratio of true positives to all positive predictions
- Recall: The ratio of true positives to all actual positives
- F-score: The harmonic mean of precision and recall

2 Introduction

The task of complex word identification aims to automatically detect words that might be difficult for readers to understand. This classification problem employs various models to predict whether a word is 'complex' (1) or 'simple' (0).
The classification approaches range from simple baselines to more sophisticated machine learning models:

1. Baseline Methods:
   - All-complex classification
   - Word length thresholding
   - Word frequency thresholding
2. Machine Learning Models:
   - Naive Bayes
   - Logistic Regression
   - Support Vector Machines (SVM)
   - Random Forest
   - Decision Trees

These models utilize features that include word length, Google NGram frequency counts, syllable count, and WordNet synset information to make predictions.

2.1 Model Evaluation Strategy

The F-score metric is my primary evaluation measure, calculated as:

$$F\ score = 2 \cdot \frac{precision \cdot recall}{precision + recall} \qquad (1)$$

In my implementation, F scores are converted to percentages (0-100%) where:

- 100% indicates perfect classification
- 0% indicates complete misclassification
- Typical scores of 60-80% suggest good model performance

We prioritize the F-score because it balances:

1. Precision: Accuracy of positive predictions
2. Recall: Ability to find all complex words

This is particularly important for complex word identification, where both false positives (incorrectly flagging simple words) and false negatives (missing complex words) have significant implications for readability assessment.

3 Report

3.1 All-Complex

The following evaluation metric was run for a classifier that just assumed that every word was complex (hence assigning a 0). This classification method should undoubtedly never be used as a means to actually classify documents, but it was interesting to see how it performed just assigning all to a complex without any other feature consideration.

```
Majority class baseline
-----------------------
Performance on training data
Accuracy: 43%
Precision: 43%
Recall: 100%
F-score: 60%

Performance on development data
Accuracy: 44%
Precision: 44%
Recall: 100%
F-score: 61%
```

The classifier performed similarly with both the training and development data sets, as expected since there is no real feature being checked.

3.2 Word Length

The evaluation metrics below reflect the performance on a classifier that used word length as it's only feature of classification. After running the evaluation on all word lengths, It can be noted that a base length threshold of 6 performed best out of all other word lengths on the training and development data sets. After running this threshold on the classification model, we can see the following evaluation metrics, showcasing an F-score of 72% on the development data.

```
Word length baseline
--------------------
Best length threshold: 6

Training data results:
Accuracy: 69%
Precision: 60%
Recall: 84%
F-score: 70%

Development data results:
Accuracy: 70%
Precision: 62%
Recall: 86%
F-score: 72%
```

3.3 Word Frequency

For the frequency baseline of the word, I decided to work with classifying words that have a frequency beyond a certain threshold as complex (1) and below simple (0). To find that threshold, I decided to make a logarithmic distribution of the range of max and min frequency words for 20 values that I tried for all and chose the one with the highest F-score upon evaluation. Below are the results

```
Maximum frequency count: 47376829651
Minimum frequency count: 40
Best frequency threshold: 416

Training data results:
Accuracy: 44%
Precision: 43%
Recall: 96%
F-score: 60%

Development data results:
Accuracy: 44%
Precision: 43%
Recall: 95%
F-score: 60%
```

As you can see from the above data, the maximum frequency for a word was 47376829651 while the minimum was 40. As a result, we can see that the best performing threshold based on our logarithmic distribution was a threshold of 416, meaning words with higher frequency were classified as complex, otherwise simple.

4 Basic Classifiers (Implementation)

4.1 Naive Bayes

This classifier is an implementation of the Naive Bayes classifier model by sklearn which uses a GaussianNB classifier model to operate. Below you can see the same evaluation metrics on the training and development data sets.

```
Naive Bayes
-----------
Training Set Metrics:
Accuracy: 55%
Precision: 49%
Recall: 98%
F-score: 65%

Development Set Metrics:
Accuracy: 55%
Precision: 50%
Recall: 98%
F-score: 66%
```

Performing slightly worse than the Word Length model, this classifier although having very high recall on both data sets, has low accuracy and precision, leading to a low f-Score, relatively speaking.

4.2 Logistic Regression

The next Classifier is the Logistic Regression model from sklearn once again. Below are the evaluation metrics from that classifier on the same data we've been working with up until now.

```
Logistic Regression
-----------
Training Set Metrics:
Accuracy: 74%
Precision: 72%
Recall: 65%
F-score: 68%

Development Set Metrics:
Accuracy: 77%
Precision: 76%
Recall: 70%
F-score: 73%
```

As we can see, this model although not performing in any outstanding feat across all metrics, has the highest F-score we've see so far with 73% on development data, making it a somewhat good classifier.

4.3 Classifier Comparison and Discussion

Comparison of all classifiers, discussing strengths and weaknesses.

5 BYO Classifier

5.1 Build Your Own - Description

5.2 Custom Classifier Features

My complex word identification system utilizes five key linguistic features:

1. **Word Length**
   - Measures character count per word
   - Longer words tend to be more complex
2. **Word Frequency**
   - Based on Google NGram counts
   - Less frequent words are often more complex
3. **Syllable Count**
   - Counts syllables using phonetic rules
   - Multi-syllabic words correlate with complexity
4. **WordNet Synsets**
   - Counts number of word meanings
   - Words with more meanings can indicate complexity

These features are combined as input vectors for various classifiers (Random Forest, Decision Tree, and SVM) to determine word complexity.

5.2.1 Classifier

For my own built classifier, I decided to test 3 different classifier models from the sklearn library and compare each of them to see which one performed best on the F-score metric. The 3 classifiers I decided to compare were RandomForestClassifier, DecisionTreeClassifier and SVC. I ran comparison metrics on each of them the same way I ran them on prior classifiers to keep my methodology consistent with the same training and development data. I picked those classifiers because of their popularity and my intrigue after they were mentioned in the instructions.

5.3 Build Your Own - Performance

Performance analysis, including accuracy, precision, and recall.

To pick on the appropriate classifier for me to use, I decided to run the same parameters for each (e.g random_state=20) and look at their evaluation metrics and chose the one with the highest F-score. I chose 20 as an arbitrary number to keep the randomness constant across all the experiments ran. Doing so gave me the following results:

```
== Classifier: RandomForestClassifier ==
 Training Set Metrics:
 Accuracy: 99%
 Precision: 99%
 Recall: 98%
 F-score: 99%

 Development Set Metrics:
 Accuracy: 76%
 Precision: 72%
 Recall: 72%
 F-score: 72%

== Classifier: DecisionTreeClassifier ==
 Training Set Metrics:
 Accuracy: 99%
 Precision: 99%
 Recall: 98%
 F-score: 99%

 Development Set Metrics:
```

```
Accuracy: 70%
Precision: 67%
Recall: 64%
F-score: 65%

======== Classifier: SVC ========
Training Set Metrics:
Accuracy: 74%
Precision: 70%
Recall: 71%
F-score: 70%

Development Set Metrics:
Accuracy: 77%
Precision: 74%
Recall: 74%
F-score: 74%
```

Considering all the above results, I decided to choose the SVC model purely because it had a higher F-score value for the development data set than the others. This leads me to think that this classifier model performs better on unseen data.

5.4 Build Your Own - Error Analysis

Examination of errors, misclassifications, and potential improvements.

The Support Vector Machine (SVM) classifier implemented for my classifier uses four key features:

- Word length
- Word frequency
- Syllable count
- Number of synonyms

Analysis of the classification results shows several interesting patterns in both correct and incorrect predictions:

5.4.1 Misclassification Patterns

Common words were sometimes incorrectly classified as complex due their length overwhelming their frequency count due to being plurals like the following examples:

```
potentially: 1
shoppers: 1
firearms: 1
```

Those examples were most likely wrongly classified due to their length being beyond the threshold of 6.

5.4.2 Successful Classifications

The classifier however did show signs of correctly identifying complex words, like the following:

```
sargassum: 1
stalagmites: 1
biosphere: 1
pediatricians: 1
subway: 0
full: 0
```

Simple Words like "subway" and "full" were correctly classified based on:

- High frequency in text usage
- Shorter length
- Lower syllable count
- High synonyms count

While the actual complex words such as "sargassum", "stalagmites" etc. were most likely based on:

- Low frequency in text usage
- Longer length beyond the threshold
- Higher syllable count
- Low synonyms count

5.4.3 Areas for Improvement

The classification model could be improved by:

- Better handling of variant words such as plurals
- Revised weighting of features to prevent length/syllable count from dominating
- Incorporating additional features for common word variations
- Larger data set to improve pattern recognition by the model