COM6513 NLP Class project: Complex Word Identification

Xinghui He
University of Sheffield , United Kingdom
xhe15@sheffield.ac.uk

Abstract

This paper is for COM6513 module's class project at the University of Sheffield attempt the Complex Word Identification Shared Task 2018. In this class project, we will be tackling the binary classification task for the monolingual English and monolingual Spanish tracks with ensemble classifiers to distinguish complex and non-complex words.

1 Introduction

The task of Complex Word Identification(cwiSharedTask, 2018) is to identify which words in reading texts are difficult words for some specific readerships to understand. It is challenging because different people may have totally different kinds of barriers on the way of understanding texts. To help different people, like non-native speakers, native speakers but with low literacy levels or reading disabilities, to understand texts better, the step of Complex Word Identification plays an important role in building text simplification systems to facilitate reading comprehension.

2 Methods and Experiments

To develop a binary classification complex word identification system, firstly, we can combine nltk's POS-tagger, lemmatizer function with sklearn's libraries or regex tools to preprocessing the text data. And then extract new features(such as word frequency, n-gram, pos, word2vec) with sklearn, nltk and gensim tools. gensim's word2vec library offers an surprisingly well-working and swift AI-approach to extract raw texts' semantic features, based on a shallow neural network. At last, it is important to implement sklearn's efficient machine learning frameworks to binary classify the complex words and non-complex words. With evaluating the performance of development dataset and test dataset, we can learn to optimize the parameter for suitable machine learning models of English dataset, Spanish dataset respectively in order to build a good CWI system.

2.1 Data

The English dataset origins from professionally written news, non-professionally written news (WikiNews), and Wikipedia articles. While the Spanish dataset only collects from some Wikipedia pages in Spanish. The origins plays an vital role of deciding texts' topic, way of presentation and even the number of complex words. For example, texts from professionally written news may present more formal word instead of some similar informal or spoken words from Wikipedia. And furthermore, there are usually more complex words in professionally written texts since it has high requirement for academic language ability and also the base of culture knowledge. Thus, we can infer that our English dataset has a wide range differ with our Spanish dataset. Therefore, I would implement different sklearn's machine learning model with English dataset and Spanish dataset respectively, to get better performance for both of them.

In terms of rules of annotating labels, the way in which human annotate words to labels effect a lot on CWI system's performance(Marcos Zampieri1, 2017). In our English dataset, each sentence was annotated by 10 native speakers and 10 non-native speakers and they would label the word according to the surrounded context to decide if the word is difficult for non-native speakers, low literacy based or unable reading native speakers. Due to different culture and language ability level, native speakers and non-native speakers may differ in the opinion of which word should be labeled as "com-

Language	training set	dev set	test set
English	27,299	3,328	4,252
Spanish	13,750	1,622	2,233

Table 1: Data Instances

plex". Therefore, to make the results of labeling more reasonable, it is necessary to get similar amount of native and non-native annotators. However, our Spanish dataset was annotated by only 10 people mixed with native and non-native speakers. And worse more, by executing code

```
print(sent\['native\_annots'\])
```

and

```
print(sent\['nonnative\_annots'\])
```

in our program, we can see that most of the data was annotated by 10 native speakers, only a few sentences were annotated by 1 or 2 non-native speakers. Therefore, this may cause a reliability problem with our golden label.

2.2 Baseline system

As Figure 1 shows, our baseline use macro-F1 score, as balanced F-score or F-measure, to evaluate the task performance. It would also be the evaluation method for our Improved system. Because it calculates metrics for each label, and find their unweighted mean. The most important is that this method does not take label imbalance into account. And this is exactly suitable for our task since the number of complex words and non-complex words varies a lot. In this system, compared with original baseline system, the only thing i changed it is the sklearn machine learning module. The original model is sklearn's Logistic Regression model, it is already a good classify model to deal with our task. I replace it with a svm classifier when dealing with English track, the Spanish track still use the Logistic Regression model. And this makes the performance of english track increases 2% with original len_chars and len_tokens feature.

2.3 Improved system

Along with the rapid development of machine learning or deep learning, there are various of ways and tools to tackling the task and to improve the baseline performance. In this machine learning task, to achieve the scores of developing such a system, there are a few important steps.

Figure 1: baseline f1-score on devset

```
baseline_cwi
/Users/xinghuihe/anaconda/bin/python
english: 27299 training - 3328 dev
macro-F1: 0.71

spanish: 13750 training - 1622 dev
macro-F1: 0.72
```

Firstly, we need to select informative features, and sklearn's cross_val_score function is helpful to identify which feature is more useful in some specific tasks since we can get and compare different scores when changing features. And secondly, we need to select suitable method to evaluate system. In our task this time, macro-F1 is good with a text binary classify problem. Thirdly, selecting suitable classifier and optimize algorithm is necessary. In our improved system, i developed a small program named model_selection.py, which list a lot of supervised learning's classification api to implement with the system. We can update the parameters and select the right model through cross validation. And at last, adjusting the right model's parameters for your task and evaluate the system. In the process of developing my

Figure 2: how model_selection.py work

```
Accuracy: 0.72 (+/- 0.00) [Logistic Regression]
Accuracy: 0.73 (+/- 0.00) [Random Forest]
Accuracy: 0.68 (+/- 0.01) [GaussianNB]
Accuracy: 0.73 (+/- 0.00) [Ensemble1]
Accuracy: 0.73 (+/- 0.00) [Decision Tree]
Accuracy: 0.68 (+/- 0.04) [KNeighbors]
Accuracy: 0.74 (+/- 0.00) [SVC]
```

Improved system, i find that sklearn's VotingClassifier is interesting. Not like the single classifier, it need combine some classifier together by yourself. I think it may be great for some experienced machine learning learner to do it. Because each classifier has its characteristic, experienced learner may combine them in a beautiful way.

3 Results

The results are even a little bit worse than the baseline system. I think it maybe the problem of extracting feature. It did not implement really right in my code. And that even ruin the original feature. I generated the part-of-speech label feature, word frequency feature, but maybe they are not

generated in a right way and i am too slow to fix it.

Figure 3: improved system of english track:dev and test

```
english: 27299 training - 4252 test
DEV result:
macro-F1: 0.70
  Label   Precision   Recall      F1      Support
-------------------------------------------------
    0        0.78       0.69      0.73      1940
    1        0.63       0.72      0.67      1388

TEST result:
macro-F1: 0.71
  Label   Precision   Recall      F1      Support
-------------------------------------------------
    0        0.77       0.73      0.75      2465
    1        0.65       0.70      0.67      1787
```

Figure 4: improved system of spanish track:dev and test

```
spanish: 13750 training - 2233 test
DEV result:
macro-F1: 0.70
  Label   Precision   Recall      F1      Support
-------------------------------------------------
    0        0.74       0.83      0.78       969
    1        0.69       0.56      0.62       653

TEST result:
macro-F1: 0.70
  Label   Precision   Recall      F1      Support
-------------------------------------------------
    0        0.73       0.83      0.78      1326
    1        0.70       0.56      0.62       907
```

4 Conclusion and Future Work

During this task, i think i failed in extracting features and that is a vital step in develop cwi system. In the future, i should focus more on coding with extracting more informative features, like use gensim's word2vec library or even tensorflow, pytorch to extarct semantic features and also can think more about extracting morphological and lexical features.

Figure 5: Learning curves of english track

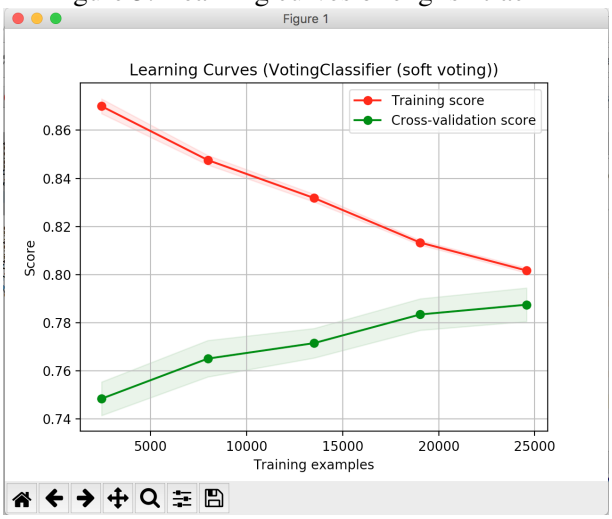
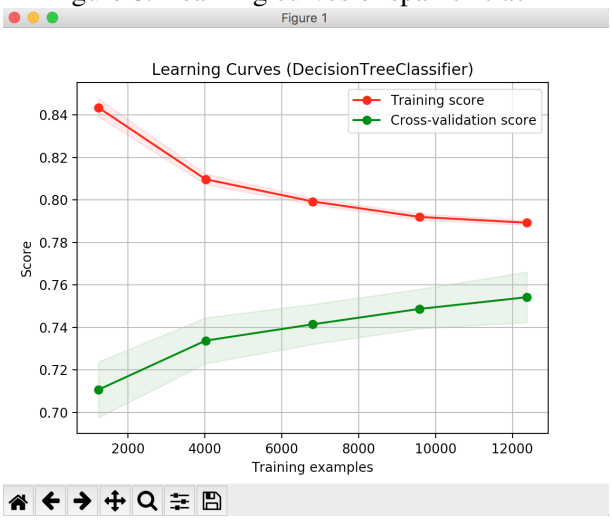


Figure 6: Learning curves of spanish track



References

cwiSharedTask. 2018. Complex word identification (cwi) shared task 2018. https://sites.google.com/view/cwisharedtask2018/home?authuser=0.

Shervin Malmasi2 Gustavo Paetzold3 Lucia Specia3 Marcos Zampieri1. 2017. *Complex Word Identification: Challenges in Data Annotation and System Performance*. Proceedings of the 4th Workshop on Natural Language Processing Techniques for Educational Applications, Taipei, Taiwan.