Construction type assignment

By: Shubham Deshmukh

Objective

To classify construction description into respective class (construction code).

Multi-class/single label classification problem.

Observations

1. 6 classes (1,2,3,4,5,6) are present that seem much similar to each other.
2. Sentences length are ~ 4-5 tokens on average
3. Imbalance data found highly biased to class '4' and '2' (combined to ~65% of total data)
4. Most frequent tokens (words) are common (present) throughout all classes like wood, brick, metal, steel, frame, concrete, and fire.
5. Training data is small in size

Approaches

1. Since sentences are short (~4 words on average) long-term dependency will not help. TF-IDF + classifier will be the first approach.
2. Frequent occurring tokens are common throughout other classes hence word-level tf-idf alone won't help. A combination of words and character can help with word boundary.
3. Since training data is ~ 1k any classifier can work (SVC) , no need to introduce the neural net that will overfit on the train set and not generalize well.

Parameters

1. High_level function:

   model=grid_searchcv(pipeline(feature_union(word level tf-idf + char_wb_level tf-idf),svc()),dictionary_of_params))

2. Gridsearch is used to find optimal hyper-parameters.
3. Parameters selected:

   {'clf__C': 1, 'tfidf__char__max_features': 500, 'tfidf__char__ngram_range': (1, 3), 'tfidf__char__stop_words': None, 'tfidf__word__max_features': 900, 'tfidf__word__ngram_range': (1, 4), 'tfidf__word__stop_words': None}

Problems

1. Imbalance data: Training data is highly biased.

Class weights: Class weights can modify the loss function directly by giving a penalty to the classes with different weights assigned. It will increase the power of the minority class and reducing the power of the majority class. It can be also referred to as Cost-sensitive learning for imbalanced classification which is focused on first assigning different costs to the types of misclassification errors that can be made.

Solution: Provide weights for each class or maintain class_weights='balanced'

Problems

2. Imbalance training and validation splitting:

    We cannot simply train-test to split the data because a random split totally disregards the distribution or proportions of the classes provided. We will end up with a train and a test set with totally different data distributions.A model that is trained on a vastly different data distribution than the test set will perform poorly at validation.

Solution = stratification while splitting

Problems

3. Since Data is highly biased to the top 2-3 classes model will keep performing better for those classes and poor for other classes even if class weights are provided.

Solution = We can implement a threshold for each class. This will increase our recall.

For biased classes like 4,2 and 3, the model will keep predicting 4,2 most of the time. If the model threshold (decision function) is not confident and has less threshold then let's not ask it to guess rather send it to NHI (Needed human intervention). 5% of the time NHI will get the hit. We can again build another model to classify ambiguous construction descriptions. This improves the whole system.

Notebooks guideline (code)

Files available:

Three files are attached.

1. Construction_type_assignment_explorer = ipython notebook that includes every single stuff implemented. This can be used for hands-on and further improvements.

2. Construction_type_assignment_module_level_code = Module wise notebook that can be used to train data in production incase.

3. Construction_type_assignment_production_level_code = Production level code that can be used directly in production. As per the PEP8 python code guide checker.