NLP

Saturday, 11 January 2025 10:02 AM

```
from sklearn import feature_extraction, linear_model, model_selection, preprocessing
```

We'll use scikit-learn's `CountVectorizer` to count the words in each tweet and turn them into data our machine learning model can process.

```
count_vectorizer = feature_extraction.text.CountVectorizer()

## let's get counts for the first 5 tweets in the data
example_train_vectors = count_vectorizer.fit_transform(train_df["text"][0:5])

## we use .todense() here because these vectors are "sparse" (only non-zero elements are kept to save space)
print(example_train_vectors[0].todense().shape)
print(example_train_vectors[0].todense())
```

Now let's create vectors for all of our tweets.

```
train_vectors = count_vectorizer.fit_transform(train_df["text"])

## note that we're NOT using .fit_transform() here. Using just .transform() makes sure
# that the tokens in the train vectors are the only ones mapped to the test vectors -
# i.e. that the train and test vectors use the same set of tokens.
test_vectors = count_vectorizer.transform(test_df["text"])

## Our vectors are really big, so we want to push our model's weights
## toward 0 without completely discounting different words - ridge regression
## is a good way to do this.
clf = linear_model.RidgeClassifier()
```

##Let's test our model and see how well it does on the training data. For this we'll use `cross-validation` - where we train on a portion of the known data, then validate it with the rest. If we #do this several times (with different portions) we can get a good idea for how a particular model or #method performs.

The metric for this competition is F1, so let's use that here.

```
scores = model_selection.cross_val_score(clf, train_vectors, train_df["target"], cv=3, scoring="f1")
scores
```

let's do predictions on our training set and build a submission for the competition.

```
clf.fit(train_vectors, train_df["target"])
```

```
RidgeClassifier(alpha=1.0, class_weight=None, copy_X=True, fit_intercept=True,
                max_iter=None, normalize=False, random_state=None,
                solver='auto', tol=0.001)
```