AI Search

A state machine used to load data from the database and to train, test and store a network.

The network is represented as a pipeline containing TokenCountVectorizer, TF-IDFTransformer and a NaiveBayes classifier.

The network is stored in memory using the ModelManager.

This is the only class that uses with php-ml.

Get State

To get the state of AI Search, use `getState`. The possible states are `"nothing"`, `"loaded"`, `"vectorized"` and `"trained"`. The current state will be loaded from memory during construction.

```
$ai = new AISearch;
if($ai->getState() == "nothing")
{
    echo "ready to read.";
}
else if($ai->getState() == "trained")
{
    echo "ready to test.";
}
```

Set Data

Meta data for the storage and network can be set through `setStorageData` and `setNetworkData` respectively. An object implementing `Data` must be passed and is set if valid.

```
$ai = new AISearch;
$ai->setStorageData($storageData); //works if($storageData->isValid())
$ai->setNetworkData($networkData); //works if($networkData->isValid())
```

Train and Test Network

To retrive, train on data and test the network, use `trainAndTestNetwork` with parameters `NetworkData` and `StorageData` the default values of which are `null`.

The returned `float` is the accuracy of the network, `0` being completely inaccurate and `1` being totally accurate.

```
$ai = newAISearch;
$ai->trainAndTestNetwork();
//returns accuracy as float 0-1
```

Predict

To predict what course a query will return, use the `predict` function. This function loads the network from memory (if not already loaded) and returns the prediction for the given input. To load the network ahead of time, preceed `predict` with `primePredictor`.

```
$ai = new AISearch;
$ai->primePredictor(); //not necassary but reduces time for predict to execute.
$ai->predict($query);
//returns course ID of prediction for $query
```

Read Data

To read and store data from the database, use `readData`. If the state of AI Search is `"nothing"`, course data is read from a database using [`CourseRetriver`]() and stored under `data/*courseID*/`. Finally, state is set to `"loaded"`.

```
$ai = new AISearch;
if($ai->getState() == "nothing")
{
    $ai->readData();
    //$ai->getState() returns "loaded"
}
```

DoTFIDF

To do the tf-idf seperately from training, use `doTFIDF`. If the state of AI Search is `"loaded"`, the data will be vectorized and stored in `samples/raw`.

```
$ai = new AISearch;
if($ai->getState() == "loaded") $ai->doTFIDF();
```

Train Network

Vectorizing and Training

To train the network on stored data, use `trainNetwork`. If the state of AI Search is `"loaded"`, data is split into training and testing sets based on the `storedData->trainingPercentage`. Training set is used to train, testing set is stored for testing at `samples/`. Finally, state is set to `"trained"`.

Training involves Tokenization of data with NGramTokenizer and english stop words, before being transformed into TF-IDF vectors and used to train a NaiveBayes network.

```
$ai = new AISearch;
if($ai->getState == "loaded")
{
    $ai->trainNetwork();
    //$ai->getState() returns "trained"
}
```

Pre-Vectorized Training

Alternatively, if the state of AI Search is `"vectorized"`, `trainNetwork` will retrieve the stored samples and train with those. The network stored still contains the necassary transformers for prediction from string.

```
$ai = new AISearch;
if($ai->getState() == "vectorized")
{
    $ai->trainNetwork();//trains with stored, vectorized samples
}
else if($ai->getState() == "loaded")
{
    $ai->trainNetwork();//uses php-ml/pipeline to vectorize and train in one
}
```

Test Network

To test the network on stored testing samples, use `testNetwork`. If the state of AI Search is `"trained"`, the method will read testing samples from file and test the network, returning accuracy as a `float` in `(0,1)`.

```
$ai = new AISearch;
if($ai->getState() == "trained")
{
    echo "Accuracy: " . $ai->testNetwork() . PHP_EOL;
}
```