**https://www.bitcard.org/register?bc\_ir=email%2Cusername&bc\_v=4&bc\_t=315ecb2e3f9493552fde42fc09ba4e&bc\_io=name&bc\_r=https%3A%2F%2Frt.cpan.org%2Findex.html**

package Text::Mining;

use base qw(Text::Mining::Base);

use Class::Std;

use Class::Std::Utils;

use Text::Mining::Corpus;

use Text::Mining::Corpus::Document;

use Text::Mining::Shell;

use warnings;

use strict;

use Carp;

use version; our $VERSION = qv('0.0.8');

{

my %attribute\_of : ATTR( get => 'attribute', set => 'attribute' );

sub BUILD {

my ($self, $ident, $arg\_ref) = @\_;

# &DBConnect( %{ $self->\_library\_connect\_parameters() } );

return;

}

sub shell { my $shell = Text::Mining::Shell->new(); $shell->cmdloop(); }

sub version { return "VERSION $VERSION"; }

sub create\_corpus { my ( $self, $arg\_ref ) = @\_; return Text::Mining::Corpus->new( $arg\_ref ); }

sub get\_corpus { my ( $self, $arg\_ref ) = @\_; return Text::Mining::Corpus->new( $arg\_ref ); }

sub delete\_corpus { my ( $self, $arg\_ref ) = @\_; my $corpus = Text::Mining::Corpus->new(); return $corpus->delete( $arg\_ref ); }

sub get\_root\_dir { my ( $self ) = @\_; return $self->\_get\_root\_dir(); }

sub get\_root\_url { my ( $self ) = @\_; return $self->\_get\_root\_url(); }

sub get\_data\_dir { my ( $self, $corpus\_id ) = @\_; return $self->\_get\_data\_dir( $corpus\_id ); }

sub get\_submitted\_document { my ( $self, $arg\_ref ) = @\_; return Text::Mining::Corpus::Document->new( $arg\_ref ); }

sub count\_submitted\_waiting { my ( $self ) = @\_; my ( $count ) = $self->library()->sqlexec( "select count(\*) from submitted\_documents where exit\_date = '0000-00-00 00:00:00'", '@' ); return $count; }

sub count\_submitted\_complete { my ( $self ) = @\_; my ( $count ) = $self->library()->sqlexec( "select count(\*) from submitted\_documents where exit\_date != '0000-00-00 00:00:00'", '@' ); return $count; }

sub parse\_document {

my ( $self, $arg\_ref ) = @\_;

my $document = defined $arg\_ref->{document} ? $arg\_ref->{document} : $self->\_status( "No document to parse." );

my $algorithm = defined $arg\_ref->{algorithm} ? $arg\_ref->{algorithm} : $self->\_status( "No algorithm defined." );

return $document;

}

sub get\_all\_corpuses {

my ( $self, @corpuses) = @\_;

my $corpuses = $self->library()->sqlexec( "select corpus\_id from corpuses order by corpus\_id asc", '\@@' );

foreach my $corpus (@$corpuses) { push @corpuses, Text::Mining::Corpus->new({ corpus\_id => $corpus->[0] }); }

return \@corpuses;

}

sub get\_corpus\_id {

my ( $self, $arg\_ref ) = @\_;

my $corpus = Text::Mining::Corpus->new();

my ( $corpus\_id ) = $self->library()->sqlexec( "select corpus\_id from corpuses where corpus\_name = '" . $arg\_ref->{corpus\_name} . "'", '@' );

return $corpus\_id;

}

sub process\_urls {

my ( $self ) = @\_;

my $corpuses = $self->get\_all\_corpuses();

foreach my $corpus( @$corpuses ) {

my $data\_dir = $self->get\_data\_dir( $corpus->get\_id() );

my $sql = "select submitted\_url\_id, corpus\_id, submitted\_by\_user\_id, submitted\_url from submitted\_urls where exit\_date = '0000-00-00 00:00:00' and file\_not\_found = 0";

my $urls = $self->library()->sqlexec( $sql, '\@@' );

foreach my $url ( @$urls ) { $self->\_download\_url( $url, $data\_dir ); }

}

}

sub reprocess\_urls {

my ( $self ) = @\_;

my $corpuses = $self->get\_all\_corpuses();

foreach my $corpus( @$corpuses ) {

my ( $corpus\_id ) = @$corpus;

my $data\_dir = $self->get\_data\_dir( $corpus->get\_id() );

my $sql = "select submitted\_url\_id, corpus\_id, submitted\_by\_user\_id, submitted\_url from submitted\_urls where file\_not\_found = 1";

my $urls = $self->library()->sqlexec( $sql, '\@@' );

foreach my $url ( @$urls ) { $self->\_download\_url( $url, $data\_dir ); }

}

}

sub \_download\_url {

my ( $self, $url\_row, $data\_dir ) = @\_;

my ( $id, $corpus\_id, $user\_id, $url ) = @$url\_row;

my $file\_name = $self->\_parse\_file\_name( $url );

my $path = $self->\_build\_directories( $url, $data\_dir );

my $bytes = $self->\_download\_file({ target\_dir => $data\_dir . $path,

url => $url,

file\_name => $file\_name });

if ( $bytes ) {

my $sql = "insert into submitted\_documents (submitted\_url\_id, corpus\_id, submitted\_by\_user\_id, document\_path, document\_file\_name, bytes ) ";

$sql .= "values ('$id', '$corpus\_id', '$user\_id', '$path', '$file\_name', '$bytes' )";

$self->library()->sqlexec( $sql );

$self->library()->sqlexec( "update submitted\_urls set exit\_date = now(), file\_found = 1, file\_not\_found = 0 where submitted\_url\_id = '$id'" ); }

else {

$self->library()->sqlexec( "update submitted\_urls set exit\_date = now(), file\_found = 0, file\_not\_found = 1 where submitted\_url\_id = '$id'" ); }

}

sub \_build\_directories {

my ( $self, $url, $corpus\_data\_dir ) = @\_;

my @path = split(/\//, $url); shift(@path); shift(@path); # Remove protocol

my $file = pop(@path);

my $path = '';

foreach my $part (@path) {

$path .= '/' . $part;

if (! -e $corpus\_data\_dir . $path) { mkdir $corpus\_data\_dir . $path; }

}

return $path;

}

}

1; # Magic true value required at end of module

\_\_END\_\_