BAB 4. SEGMENTASI PROGRAM

Bab ini akan membahas implementasi sistem berdasarkan desain sistem di Bab 3. Akan ditampilkan segmen-segmen program untuk metode *stemming* yang digunakan, proses preparasi data atau *preprocessing*, *indexing*, *searching*, dan pengaksesan basis data dengan REST.

4.1. Naskah PHP Proses *Preprocessing*

Proses *preprocessing* ini dimulai dengan penghapusan karakter-karakter spesial, kemudian dilakukan pengecekan bahasa yang digunakan abstrak tersebut. Bila abstrak berbahasa Indonesia, maka abstrak akan dilewati atau tidak dianggap. Bila abstrak berbahasa Inggris, maka akan dilakukan penghapusan *stopwords*. Kemudian akan dilakukan *stemming*.

Daftar *stopwords* yang digunakan didapat dari *website* https://github.com/igorbrigadir/stopwords/blob/master/en/nltk.txt.

Segmen Program 4.1. Proses *Preprocessing*

```
<?php
include "porter.php";

	function cleanString($string) {
		$res = strip_tags($string,"");
		$res = utf8_decode($res);
		$specialchar = array("-","/","-", ".",
"?","\r","\n","\t","'");
		$res = str_replace($specialchar, " ", $res);
		$res = preg_replace("/[^a-zA-Z0-9\s]/", "", $res);
		$res = strtolower($res);
		return $res;
	}

	function checkLanguage($string) {
		if (strpos($string, "abstract in bahasa indonesia")) {
			$res=explode("abstract in bahasa indonesia",
$string);

			$abstract_en = $res[0];
			$abstract_ind = $res[1];

			if (strpos($abstract_ind, "kata kunci")) {
```

Segmen Program 4.1. Proses *Preprocessing* (sambungan)

```
$res_ind=explode("kata kunci", $abstract_ind);
                        $abstract_ind = $res_ind[0];
                  }
                  return $abstract_en;
            } else {
                  if (strpos($string, "sebuah") or strpos($string,
"adalah") or strpos($string, "dalam" or strpos($string,
"penelitian"))) {
                        if (strpos($string, "abstract in bahasa
indonesia")) {
                              str_replace('abstract in bahasa
indonesia', '', $string);
                        }
                        return NULL;
                  } else {
                        return $string;
                  }
            }
      }

      function stopWordsRemoval($string) {
      $stopwords=array("i","me","my","myself","we","our","ours","o
urselves","you","your","yours","yourself","yourselves","he","him",
"his","himself","she","her","hers","herself","it","its","itself","
they","them","their","theirs","themselves","what","which","who","w
hom","this","that","these","those","am","is","are","was","were","b
e","been","being","have","has","had","having","do","does","did","d
oing","a","an","the","and","but","if","or","because","as","until",
"while","of","at","by","for","with","about","against","between","i
nto","through","during","before","after","above","below","to","fro
m","up","down","in","out","on","off","over","under","again","furth
er","then","once","here","there","when","where","why","how","all",
"any","both","each","few","more","most","other","some","such","no"
,"nor","not","only","own","same","so","than","too","very","s","t",
"can","will","just","don","should","now","d","ll","m","o","re","ve
","y","ain","aren","couldn","didn","doesn","hadn","hasn","haven","
isn","ma","mightn","mustn","needn","shan","shouldn","wasn","weren"
,"won","wouldn");
            $array = explode(" ", $string);
            $array = array_diff($array,$stopwords);
            $string = implode(" ", $array);
            return $string;
      }

      function preProcessing($string) {
            $string=cleanString($string);
            $res = checkLanguage($string);
            if (!is_null($res)) {
      $res = stopWordsRemoval($res);
```

Segmen Program 4.1. Proses *Preprocessing* (sambungan)

```
                  $res = explode(" ", $res);
                  $valueArr=array();
                  foreach ($res as $key => &$value) {
                        $temp=$value;
                        $value=PorterStemmer::stem($value);

                        array_push($valueArr,$value);
                  }
                  $res = implode(" ", $valueArr);
                  return $res;
            } else {
                  return NULL;
            }
      }
?>
```

4.2. Naskah PHP Proses *Stemming*

Berikut adalah naskah dari algoritma Porter *Stemming* dan *New* Porter *Stemming*.

4.2.1. Naskah PHP Porter *Stemmer*

Naskah PHP Porter *Stemmer* yang digunakan diambil dari *website* resmi Porter *Stemmer*, yaitu https://tartarus.org/martin/PorterStemmer/php.txt.

Segmen Program 4.2. Porter *Stemmer*

```
<?php

      class PorterStemmer {
            public static $regex_consonant =
'(?:[bcdfghjklmnpqrstvwxz]|(?<=[aeiou])y|^y)';

            public static $regex_vowel =
'(?:[aeiou]|(?<![aeiou])y)';

            public static function cvc($str) {
                  $c=self::$regex_consonant;
                  $v=self::$regex_vowel;
                  return preg_match("#($c$v$c)$#", $str,$matches)
and strlen($matches[1])==3 and $matches[1]{2}!='w' and
$matches[1]{2}!='x' and $matches[1]{2}!='z';
            }

            public static function doubleConsonant($str) {
                  $c=self::$regex_consonant;
                  return preg_match("#($c{2}$)#", $str, $matches)
and $matches[0]{0}==$matches[0]{1};
            }
```

Segmen Program 4.2. Porter *Stemmer* (sambungan)

```
            public static function m($str) {
                  $c=self::$regex_consonant;
                  $v=self::$regex_vowel;
                  $str=preg_replace("#^$c+#", "", $str);
                  $str=preg_replace("#$v+$#", "", $str);
                  preg_match_all("#($v+$c+)#", $str, $matches);
                  return count($matches[1]);
            }

            public static function replace(&$str, $check, $replac,
$m=NULL) {
                  $len=0-strlen($check);
                  if (substr($str,$len)==$check) {
                        $substr=substr($str,0,$len);
                        if (is_null($m) OR self::m($substr)>$m) {
                              $str = $substr.$replac;
                        }
                        return true;
                  }
                  return false;
            }

            //Step 1
            public static function step1ab($word) {
                  //Step 1a
                  if (substr($word, -1)=='s') {
                        self::replace($word, "sses", "ss")
                        OR self::replace($word, "ies", "i")
                        OR self::replace($word,"ss","ss")
                        OR self::replace($word,"s","");
                  }
                  //Step 1b
                  if (substr($word, -2,1) != "e" or
!self::replace($word,"eed","ee",0)) {
                        $v=self::$regex_vowel;
                        if (preg_match("#$v+#", substr($word,0,-
3)) && self::replace($word,'ing', '') OR preg_match("#$v+#",
substr($word,0,-2) && self::replace($word,'ed',''))) {
                              if (!self::replace($word,'at','ate')
AND !self::replace($word,'bl','ble') AND
!self::replace($word,'iz','ize')) {
                                    if
(self::doubleConsonant($word) AND substr($word,-2)!='ll'
AND substr($word,-2)!='ss'
AND substr($word,-2)!='zz') {
      $word=substr($word, 0,-1);
      } else if (self::m($word)==1 AND self::cvc($word)) {
                                          $word.='e';
                                    }
```

Segmen Program 4.2. Porter *Stemmer* (sambungan)

```
                              }
                        }
                  }
                  return $word;
            }

            //Step 1c
            public static function step1c($word) {
                  $v=self::$regex_vowel;
                  if (substr($word, -1) == 'y' &&
preg_match("#$v+#", substr($word,0,-1))) {
                        self::replace($word,'y','i');
                  }
                  return $word;
            }

            public static function step2($word) {
                  switch (substr($word, -2,1)) {
                        case 'a':

      self::replace($word,'ational','ate',0) or
self::replace($word,'tional','tion',0);
                              break;

                        case 'c':
                              self::replace($word,'enci','ence',0)
or self::replace($word, 'anci','ance',0);
                              break;

                        case 'e':
                              self::replace($word,'izer','ize',0);
                              break;

                        case 'g':
                              self::replace($word,'logi','log',0);
                              break;

                        case 'l':
                              self::replace($word,'bli','ble',0)
or self::replace($word,'alli','al',0) or
self::replace($word,'entli','ent',0) or
self::replace($word,'eli','e',0) or
self::replace($word,'ousli','ous',0);
                              break;

                        case 'o':
      self::replace($word,'ization','ize',0) or
self::replace($word,'ation','ate',0) or
self::replace($word,'ator','ate',0);
```

Segmen Program 4.2. Porter *Stemmer* (sambungan)

```
                              break;

                        case 's':
                              self::replace($word,'alism','al',0)
or self::replace($word,'iveness','ive',0) or
self::replace($word,'fulness','ful',0) or
self::replace($word,'ousness','ous',0);
                              break;

                        case 't':
                              self::replace($word,'aliti','al',0)
or self::replace($word,'iviti','ive',0) or
self::replace($word,'biliti','ble',0);
                  }
                  return $word;
            }

            public static function step3($word) {
                  switch (substr($word, -2,1)) {
                        case 'a':
                              self::replace($word,'ical','ic',0);
                              break;

                        case 's':
                              self::replace($word,'ness','',0);
                              break;

                        case 't':
                              self::replace($word,'icate','ic',0)
or self::replace($word,'iciti','ic',0);
                              break;

                        case 'u':
                              self::replace($word,'ful','',0);
                              break;

                        case 'v':
                              self::replace($word,'ative','',0);
                              break;

                        case 'z':
                              self::replace($word,'alize','al',0);
                              break;
                  }
                  return $word;
            }
            //Step 4
            public static function step4($word) {
                  switch (substr($word,-2,1)) {
```

Segmen Program 4.2. Porter *Stemmer* (sambungan)

```
                        case 'a':
                              self::replace($word,'al','',1);
                              break;

                        case 'c':
                              self::replace($word,'ance','',1) or
self::replace($word,'ence','',1);
                              break;

                        case 'e':
                              self::replace($word,'er','',1);
                              break;

                        case 'i':
                              self::replace($word,'ic','',1);
                              break;

                        case 'l':
                              self::replace($word,'able','',1) or
self::replace($word,'ible','',1);
                              break;

                        case 'n':
                              self::replace($word,'ant','',1) or
self::replace($word,'ement','',1) or
self::replace($word,'ment','',1) or
self::replace($word,'ent','',1);
                              break;

                        case 'o':
                              if (substr($word, -4)=='tion' or
substr($word, -4)=='sion') {

      self::replace($word,'ion','',1);
                              } else {

      self::replace($word,'ou','',1);
                              }
                              break;

                        case 's':
                              self::replace($word,'ism','',1);
                              break;

                        case 't':
                  self::replace($word,'ate','',1) or
self::replace($word,'iti','',1);
                              break;
                        case 'u':
```

Segmen Program 4.2. Porter *Stemmer* (sambungan)

```
                                    self::replace($word,'ous','',1);
                                    break;

                              case 'v':
                                    self::replace($word,'ive','',1);
                                    break;

                              case 'z':
                                    self::replace($word,'ize','',1);
                                    break;
                        }
                        return $word;
                  }

                  public static function step5($word) {
                        //Step 5a
                        if (substr($word, -1)=='e') {
                              if (self::m(substr($word,0,-1))>1) {
                                    self::replace($word,'e','',1);
                              } else if (self::m(substr($word,0,-1))==1)
{ //(m=1 and not *o)
                                    if (!self::cvc(substr($word,0,-1)))
{
                                          self::replace($word,'e','');
                                    }
                              }
                        }
                        //Step 5b
                        if (self::m($word)>1 and
self::doubleConsonant($word) and substr($word,-1)=='l') {
                              $word=substr($word, 0,-1);
                        }
                        return $word;
                  }

                  public static function stem($word) {
                        if (strlen($word)<=2) {
                              return $word;
                        }
                        $word = self::step1ab($word);
                        $word = self::step1c($word);
                        $word = self::step2($word);
                        $word = self::step3($word);
                        $word = self::step4($word);
                        $word = self::step5($word);
                        return $word;
                  }
            }
?>
```

4.2.2. Naskah PHP *New* Porter *Stemmer*

Naskah PHP algoritma *New* Porter *Stemmer* dijalankan sesuai dengan penjelasan pada bagian 2.5. Tetapi untuk Kasus 6 (Penanganan *Compound Suffixes*) tidak bisa dijalankan secara menyeluruh karena keterbatasan data. Sehingga untuk penanganan *compound suffixes* hanya akhiran –atization, -atist, -atism, -atic, -atical yang akan diubah menjadi akhiran –ate.

Segmen Program 4.3. *New* Porter *Stemmer*

```
<?php

      class NewPorterStemmer {
            public static $regex_consonant =
'(?:[bcdfghjklmnpqrstvwxz]|(?<=[aeiou])y|^y)';

            public static $regex_vowel =
'(?:[aeiou]|(?<![aeiou])y)';

            public static function cvc($str) {
                  $c=self::$regex_consonant;
                  $v=self::$regex_vowel;
                  return preg_match("#($c$v$c)$#", $str,$matches)
and strlen($matches[1])==3 and $matches[1]{2}!='w' and
$matches[1]{2}!='x' and $matches[1]{2}!='z';
            }

            public static function doubleConsonant($str) {
                  $c=self::$regex_consonant;
                  return preg_match("#($c{2}$)#", $str, $matches)
and $matches[0]{0}==$matches[0]{1};
            }

            public static function m($str) {
                  $c=self::$regex_consonant;
                  $v=self::$regex_vowel;
                  $str=preg_replace("#^$c+#", "", $str);
                  $str=preg_replace("#$v+$#", "", $str);
                  preg_match_all("#($v+$c+)#", $str, $matches);
                  return count($matches[1]);
```

Segmen Program 4.3. *New* Porter *Stemmer* (sambungan)

```
            }

            public static function replace(&$str, $check, $replac,
$m=NULL) {
                  $len=0-strlen($check);
                  if (substr($str,$len)==$check) {
                        $substr=substr($str,0,$len);
                        if (is_null($m) OR self::m($substr)>$m) {
                              $str = $substr.$replac;
                        }
                        return true;
                  }
                  return false;
            }

            public static function step1c($word) {
                  $v=self::$regex_vowel;
                  if (substr($word, -1) == 'i') {
                        self::replace($word,'i','y');
                  }
                  return $word;
            }
            public static function step1ab($word) {
                  if (substr($word, -4)=='feet') {
                        self::replace($word, "feet", "foot");
                  } else if (substr($word, -3)=='men') {
                        self::replace($word, "men", "man");
                  } else if (substr($word, -2)=='ci') {
                        self::replace($word, "ci", "cus");
                  } else if (substr($word, -4)=='eaux') {
                        self::replace($word, "eaux", "eau");
                  } else if (substr($word, -8)=='children') {
                        self::replace($word, "children", "child");
                  } else if (substr($word, -5)=='wives') {
                        self::replace($word, "wives", "wife");
                  } else if (substr($word, -6)=='knives') {
                        self::replace($word, "knives", "knife");
                  } else if (substr($word, -6)=='staves') {
```

Segmen Program 4.3. *New* Porter *Stemmer* (sambungan)

```
                        self::replace($word, "staves", "staff");
                    } else if (substr($word, -6)=='wolves') {
                        self::replace($word, "wolves", "wolf");
                    } else if (substr($word, -6)=='trices') {
                        self::replace($word, "trices", "trix");
                    } else if (substr($word, -4)=='mata') {
                        self::replace($word, "mata", "ma");
                    } else if (substr($word, -2)=='ei') {
                        self::replace($word, "ei", "eus");
                    } else if (substr($word, -2)=='pi') {
                        self::replace($word, "pi", "pus");
                    } else if (substr($word, -3)=='ses') {
                        self::replace($word, "ses", "s");
                    } else if (substr($word, -3)=='xes') {
                        self::replace($word, "xes", "x");
                    } else if (substr($word, -3)=='sis') {
                        self::replace($word, "sis", "s");
                    } else if (substr($word, -3)=='xis') {
                        self::replace($word, "xis", "x");
                    }

                    if (substr($word, -5)=='iedly') {
                        self::replace($word, "ly", "");
                    } else if (substr($word, -7)=='iedness') {
                        self::replace($word, "ness", "");
                    } else if (substr($word, -4)=='edly') {
                        self::replace($word, "ly", "");
                    } else if (substr($word, -6)=='edness') {
                        self::replace($word, "ness", "");
                    } else if (substr($word, -5)=='ingly') {
                        self::replace($word, "ly", "");
                    } else if (substr($word, -7)=='ingness') {
                        self::replace($word, "ness", "");
                    }

                    if (substr($word, -4)!="ssed" && substr($word, -
3)=="sed") {
                        self::replace($word,"sed","s");
```

Segmen Program 4.3. *New* Porter *Stemmer* (sambungan)

```
                  } else if (substr($word, -5)!="ssing" &&
substr($word, -4)=="sing") {
                        self::replace($word,"sing","s");
                  }

                  if (substr($word, -1)=='s') {
                        self::replace($word, "sses", "ss")
                        OR self::replace($word, "ies", "i")
                        OR self::replace($word,"ss","ss")
                        OR self::replace($word,"s","");
                  }
                  if (substr($word, -2,1) != "e" or
!self::replace($word,"eed","ee",0)) {
                        $v=self::$regex_vowel;
                        if (preg_match("#$v+#", substr($word,0,-
3)) && self::replace($word,'ing', '') OR preg_match("#$v+#",
substr($word,0,-2) && self::replace($word,'ed',''))) {
                              if (!self::replace($word,'at','ate')
AND !self::replace($word,'bl','ble') AND
!self::replace($word,'iz','ize')) {
                                    if
(self::doubleConsonant($word) AND substr($word,-2)!='ll'
AND substr($word,-2)!='ss'
AND substr($word,-2)!='zz') {
      $word=substr($word, 0,-1);
} else if (self::m($word)==1 AND self::cvc($word)) {
      $word.='e';
                                    }
                              }
                        }
                  }
                  return $word;
            }

            public static function step2($word) {
                  if (substr($word, -9)=='atization') {
                        self::replace($word,'atization','ate',0);
                  } else if (substr($word, -5)=='atist') {
```

Segmen Program 4.3. *New* Porter *Stemmer* (sambungan)

```
                        self::replace($word,'atist','ate',0);
                  } else if (substr($word, -5)=='atism') {
                        self::replace($word,'atism','ate',0);
                  } else if (substr($word, -4)=='atic') {
                        self::replace($word,'atic','ate',0);
                  } else if (substr($word, -6)=='atical') {
                        self::replace($word,'atical','ate',0);
                  }

                  switch (substr($word, -2,1)) {
                        case 'a':

      self::replace($word,'ational','ate',0) or
self::replace($word,'tional','tion',0);
                              break;

                        case 'c':
                              self::replace($word,'enci','ence',0)
or self::replace($word, 'anci','ance',0);
                              break;

                        case 'e':
                              self::replace($word,'izer','ize',0);
                              break;

                        case 'g':
                              self::replace($word,'logi','log',0);
                              break;

                        case 'l':
                              self::replace($word,'bli','ble',0)
or self::replace($word,'alli','al',0) or
self::replace($word,'entli','ent',0) or
self::replace($word,'eli','e',0) or
self::replace($word,'ousli','ous',0);
                              break;

                        case 'o':
```

Segmen Program 4.3. *New* Porter *Stemmer* (sambungan)

```
self::replace($word,'ization','ize',0) or
self::replace($word,'ation','ate',0) or
self::replace($word,'ator','ate',0);
                              break;

                        case 's':
                              self::replace($word,'alism','al',0)
or self::replace($word,'iveness','ive',0) or
self::replace($word,'fulness','ful',0) or
self::replace($word,'ousness','ous',0);
                              break;

                        case 't':
                              self::replace($word,'aliti','al',0)
or self::replace($word,'iviti','ive',0) or
self::replace($word,'biliti','ble',0);
                  }
                  return $word;
            }

            public static function step3($word) {
                  switch (substr($word, -2,1)) {
                        case 'a':
                              self::replace($word,'ical','ic',0);
                              break;

                        case 's':
                              self::replace($word,'ness','',0);
                              break;

                        case 't':
                              self::replace($word,'icate','ic',0)
or self::replace($word,'iciti','ic',0);
                              break;

                        case 'u':
                              self::replace($word,'ful','',0);
                              break;
```

Segmen Program 4.3. *New* Porter *Stemmer* (sambungan)

```
                        case 'v':
                              self::replace($word,'ative','',0);
                              break;

                        case 'z':
                              self::replace($word,'alize','al',0);
                              break;
                  }
                  return $word;
            }

            public static function step4($word) {
                  switch (substr($word,-2,1)) {
                        case 'a':
                              self::replace($word,'al','',1);
                              break;

                        case 'c':
                              self::replace($word,'ance','',1) or
self::replace($word,'ence','',1);
                              break;

                        case 'e':
                              self::replace($word,'er','',1);
                              break;

                        case 'i':
                              self::replace($word,'ic','',1);
                              break;

                        case 'l':
                              self::replace($word,'able','',1) or
self::replace($word,'ible','',1);
                              break;

                        case 'n':
                              self::replace($word,'ant','',1) or
self::replace($word,'ement','',1) or
```

Segmen Program 4.3. *New* Porter *Stemmer* (sambungan)

```
self::replace($word,'ment','',1) or
self::replace($word,'ent','',1);
                              break;

                        case 'o':
                              if (substr($word, -4)=='tion' or
substr($word, -4)=='sion') {

      self::replace($word,'ion','',1);
                              } else {

      self::replace($word,'ou','',1);
                              }
                              break;

                        case 's':
                              self::replace($word,'ism','',1);
                              break;

                        case 't':
                              self::replace($word,'ate','',1) or
self::replace($word,'iti','',1);
                              break;

                        case 'u':
                              self::replace($word,'ous','',1);
                              break;

                        case 'v':
                              self::replace($word,'ive','',1);
                              break;

                        case 'z':
                              self::replace($word,'ize','',1);
                              break;
                  }
                  return $word;
            }
```

Segmen Program 4.3. *New* Porter *Stemmer* (sambungan)

```
            public static function step5($word) {
                  if (substr($word, -1)=='e') {
                        if (self::m(substr($word,0,-1))>1) {
                              self::replace($word,'e','',1);
                        } else if (self::m(substr($word,0,-1))==1)
{
                              if (!self::cvc(substr($word,0,-1)))
{
                                    self::replace($word,'e','');
                              }
                        }
                        self::replace($word,'e','');
                  }

                  if (self::m($word)>1 and
self::doubleConsonant($word) and substr($word,-1)=='l') {
                        $word=substr($word, 0,-1);
                  } else if (self::doubleConsonant($word)) {
                        $word=substr($word, 0,-1);
                  }
                  return $word;
            }

            public static function getService($url) {
          $ch=curl_init();
          $optArray=array(CURLOPT_URL=>$url,
CURLOPT_RETURNTRANSFER=>true);
          curl_setopt_array($ch, $optArray);
          $response=curl_exec($ch);
          $resArr=json_decode($response,true);
          return $resArr;
        }
            public static function step5new($word) {

      $verb1=self::getService("http://localhost:8088/public/irregu
lar/".$word);
                  if ($verb1!=null) {
                        $word=$verb1['verb1'];
```

Segmen Program 4.3. *New* Porter *Stemmer* (sambungan)

```
                }
                return $word;
            }

            public static function stem($word) {
                if (strlen($word)<=2) {
                    return $word;
                }
                $word = self::step1ab($word);
                $word = self::step1c($word);
                $word = self::step2($word);
                $word = self::step3($word);
                $word = self::step4($word);
                $word = self::step5($word);
                $word = self::step5new($word);
                return $word;
            }
        }
?>
```

4.3. Naskah PHP Proses *Indexing*

Proses *indexing* ini dimulai dengan pengambilan data seluruh abstrak menggunakan servis yang telah dibuat. Setiap abstrak akan dilakukan *preprocessing* terlebih dahulu, kemudian tiap kata dalam abstrak akan dipecah-pecah dan disimpan. Proses terus dijalankan hingga semua kata di abstrak telah tersimpan. Kata-kata yang telah diketahui jumlah kemunculannya dan muncul di artikel dengan ID berapa saja, akan disimpan kedalam sebuah *file* (.txt).

Segmen Program 4.4. Proses *Indexing*

```
<?php
include 'preprocessing.php';

      function getService($url) {
            $ch=curl_init();
            $optArray=array(CURLOPT_URL=>$url,
CURLOPT_RETURNTRANSFER=>true);
            curl_setopt_array($ch, $optArray);
            $response=curl_exec($ch);
            $resArr=json_decode($response,true);
            return $resArr;
```

Segmen Program 4.4. Proses *Indexing* (sambungan)

```
	}
	function getIndex() {
		$collection=array();


	$resArr=getService("http://localhost:8088/public/article");

		foreach ($resArr as $row => $value) {
			if (!is_null(preProcessing($value['abstract'])))
{$collection[$value['article_id']]=preProcessing($value['abstract'
]); }
		}
		$wordSet=array();
		$articleCount=array();

		foreach ($collection as $artID => $abstract) {
			$terms = explode(" ", $abstract);
			$articleCount[$artID]=count($terms);

			foreach ($terms as $term) {
				if (!isset($wordSet[$term])) {
					$wordSet[$term]=array('df'=>0,
'postings'=>array());
				}
				if
(!isset($wordSet[$term]['postings'][$artID])) {
					$wordSet[$term]['df']++;

	$wordSet[$term]['postings'][$artID]=array('tf'=>0);
				}

	$wordSet[$term]['postings'][$artID]['tf']++;
			}
		}
		return array('articleCount' =>$articleCount,
'wordSet'=>$wordSet);
	}

	$data=getIndex();

	$serializeData=serialize($data);
	file_put_contents('indexing.txt',$serializeData);
?>
```

4.4. Naskah PHP Proses *Searching*

Proses *searching* dilakukan ketika pengguna memasukkan kata kunci. Kata kunci yang dimasukkan akan digunakan untuk melakukan perhitungan

relevansi dengan metode TF-IDF untuk mendapatkan hasil yang sesuai. Nilai *term frequency* dan *document frequency* didapat dari *file indexing* yang telah ada sebelumnya.

Segmen Program 4.5. Proses *Searching*

```
include 'porter.php';
if (!empty($_GET['arrArtId'])) {
  $arrArtId=json_decode($_GET['arrArtId']);
}

$hasil=array();
$keyword=$_GET['inputKW'];

function tfIdf($term) {
      global $index;
      global $arrArtId;

      $term=explode(" ", $term);
      $i=0;
      foreach ($term as $key => &$value) {
            $value=PorterStemmer::stem($value);
      }

      $matchArticle=array();
      $articleCount=count($index['articleCount']);

      if (isset($index['wordSet'][$qTerm])) {
            $entry=$index['wordSet'][$qTerm];

            foreach ($entry['postings'] as $artID => $posting) {
                  $cek=true;
                  if (!empty($arrArtId)) {
                        if (in_array($artID, $arrArtId)==false)
{$cek=false;}
                  }
                  if ($cek==true) {
                        if (!isset($matchArticle[$artID])) {

      $matchArticle[$artID]=intval($posting['tf'])*log10($articleC
ount/$entry['df']);
                        } else {

      $matchArticle[$artID]+=intval($posting['tf'])*log10($article
Count/$entry['df']);
                        }
                  }
            }
      } else {
      $hasil=null;
```

Segmen Program 4.5. Proses *Searching* (sambungan)

```
      }
}
arsort($matchArticle);
return $matchArticle;
$data=file_get_contents('indexing.txt');
$index=unserialize($data);
$keyword=strtolower($keyword);
$hasil=tfIdf($keyword);
```

4.5. Naskah PHP untuk mengakses basis data dengan REST

Berikut akan ditampilkan segmen-segmen program yang digunakan untuk mengakses basis data menggunakan REST. Terdiri dari *class database, article, author, journal,* term, dan TF. Serta akan ada penjelasan untuk pemanggilan kelas-kelas tersebut.

4.5.1. *Class Database*

Koneksi dengan basis data menggunakan *class Database* seperti pada Segmen Program 4.6. Melalui *class* ini dapat dilakukan pengaksesan ke dua basis data. Basis data tersebut adalah OJS, yang berisi data-data jurnal Pusat Penelitian Universitas Kristen Petra, dan OJS_tambahan yang berisi data-data yang ditambahkan untuk keperluan pembuatan aplikasi ini.

Segmen Program 4.6. *Class Database*

```
<?php
class Database {
      public $con,$con_tambahan;

      function __construct() {
            $this->con=mysqli_connect("localhost", "root", "",
"ojs") or die("Cannot connect to DB");
            $this->con_tambahan=mysqli_connect("localhost",
"root", "", "ojs_tambahan") or die("Cannot connect to DB");
      }
};
```

4.5.2. *Class Article*

Class article berfungsi untuk mengambil data di basis data yang berhubungan dengan suatu artikel, seperti dapat dilihat pada Segmen Program 4.7. Terdapat fungsi load($id) dengan parameter id untuk mengambil informasi suatu artikel berdasarkan article_idnya, getAllArticleAbstract() untuk mengambil semua data abstrak agar dapat dilakukan *indexing*, getAdvancedSearch($term,

$article_title, $journal_id, $year, $volume, $issue, $author, $issn) dengan parameter-parameter yang akan digunakan di dalam *query* pencarian, dan getAllArticleSubject() untuk mendapatkan kata kunci setiap artikel sehingga dapat digunakan untuk data rekomendasi di *typeahead*.

Segmen Program 4.7. *Class Article*

```
<?php

class Article{
	public $db;
	public $article_id,$pages,$journal_id,$journal_title;
	public $first_name,$last_name,$middle_name;
	public $issue_id, $volume,$number,$year;
	public $setting_name, $setting_value;

	public function __construct($db) {
		$this->db=$db;
	}

	public function load($id) {
		$sql="SELECT a.article_id,a.pages,a.journal_id,
aset.setting_name, aset.setting_value FROM articles a INNER JOIN
(SELECT article_id,setting_value,setting_name FROM
article_settings WHERE article_id=".$id.") as aset ON
(aset.article_id=a.article_id)";
		$res=mysqli_query($this->db->con,$sql);
		$data=mysqli_fetch_assoc($res);
		$this->article_id = $data['article_id'];
		$this->pages = $data['pages'];
		$this->journal_id = $data['journal_id'];
		$return=[];
		while ($row=mysqli_fetch_assoc($res)) {
			if ($row['setting_name']=='pub-id::doi')
{$return['doi']=$row['setting_value'];}
			else if ($row['setting_name']=='abstract')
{$return['abstract']=$row['setting_value'];}
			else if ($row['setting_name']=='subject')
{$return['subject']=$row['setting_value'];}
			else if ($row['setting_name']=='title')
{$return['title']=$row['setting_value'];}
		}
		$return['pages'] = $data['pages'];
		$return['journal_id'] = $data['journal_id'];
		$return['article_id'] = $data['article_id'];

		$sqlJurnal="SELECT setting_value FROM journal_settings
WHERE setting_name='title' and journal_id=".$this->journal_id;
		$resJurnal=mysqli_query($this->db->con,$sqlJurnal);
		$dataJurnal=mysqli_fetch_assoc($resJurnal);
```

Segmen Program 4.7. *Class Article* (sambungan)

```
      $this->journal_title=$dataJurnal['setting_value'];
            $return['journal_title']=$this->journal_title;

            $sqlAuthor="SELECT
author_id,seq,first_name,middle_name,last_name FROM authors WHERE
submission_id=".$id;
            $resAuthor=mysqli_query($this->db->con,$sqlAuthor);
            while($rowAuthor=mysqli_fetch_assoc($resAuthor)) {

      $return['author_id'][$rowAuthor['seq']]=$rowAuthor['author_i
d'];

      $return['first_name'][$rowAuthor['seq']]=$rowAuthor['first_n
ame'];

      $return['last_name'][$rowAuthor['seq']]=$rowAuthor['last_nam
e'];

      $return['middle_name'][$rowAuthor['seq']]=$rowAuthor['middle
_name'];
            }

            $sqlIssue="SELECT issue_id,access_status FROM
published_articles WHERE article_id=".$id;
            $resIssue=mysqli_query($this->db->con,$sqlIssue);
            $dataIssue=mysqli_fetch_assoc($resIssue);
            $this->issue_id=$dataIssue['issue_id'];
            $return['issue_id'] = $this->issue_id;

            $sqlIssue2="SELECT volume,number,year FROM issues
WHERE issue_id=".$this->issue_id;
            $resIssue2=mysqli_query($this->db->con,$sqlIssue2);
            $dataIssue2=mysqli_fetch_assoc($resIssue2);
            $this->volume=$dataIssue2['volume'];
            $this->number=$dataIssue2['number'];
            $this->year=$dataIssue2['year'];
            $return['volume'] = $this->volume;
            $return['number'] = $this->number;
            $return['year'] = $this->year;

            $sqlJurnalTambahan="SELECT issn,eissn FROM journal
WHERE journal_id=".$this->journal_id;
            $resJurnalTambahan=mysqli_query($this->db-
>con_tambahan,$sqlJurnalTambahan);

      $dataJurnalTambahan=mysqli_fetch_assoc($resJurnalTambahan);
            $return['issn']=$dataJurnalTambahan['issn'];
            $return['eissn']=$dataJurnalTambahan['eissn'];
```

Segmen Program 4.7. *Class Article* (sambungan)

```
      $sqlPdf="SELECT file_name, file_type, file_size,
original_file_name FROM article_files WHERE article_id=".$this-
>article_id;
            $resPdf=mysqli_query($this->db->con, $sqlPdf);

            $dataPdf=mysqli_fetch_assoc($resPdf);
            $return['file_name']=$dataPdf['file_name'];
            $return['file_type']=$dataPdf['file_type'];
            $return['file_size']=$dataPdf['file_size'];

      $return['original_file_name']=$dataPdf['original_file_name']
;

            $sqlCategory="SELECT category_id,category_name FROM
category WHERE category_id=(SELECT category_id FROM
category_journal WHERE journal_id='".$this->journal_id."')";
            $resCategory=mysqli_query($this->db-
>con_tambahan,$sqlCategory);
            $dataCategory=mysqli_fetch_assoc($resCategory);

      $return['category_name']=$dataCategory['category_name'];

            return $return;
      }

      public function getAllArticleAbstract(){
            $sql="SELECT a.article_id, aset.setting_value as
abstract FROM articles a INNER JOIN (SELECT
article_id,setting_value FROM article_settings WHERE
setting_name='abstract') as aset ON
(aset.article_id=a.article_id)";
            $res=mysqli_query($this->db->con,$sql);
            $return=array();
            while ($row=mysqli_fetch_assoc($res)) {
                  $return[]=$row;
            }
            return $return;
      }

public function getAdvancedSearch($term, $article_title,
$journal_id, $year, $volume, $issue, $author, $issn) {
      $syarat="";
            if ($term!=""){
                  $term=trim($term);
                  if (strpos($term," ")) {
                        $res=explode(" ", $term);
                        $syarat = $syarat." AND (0";
                        for ($i=0; $i <count($res) ; $i++) {
$syarat = $syarat." OR x.abstract LIKE '%".$res[$i]."%'";
```

Segmen Program 4.7. *Class Article* (sambungan)

```
      }
                        $syarat = $syarat." )";
                  } else {
                        $syarat = $syarat." AND x.abstract LIKE
'%".$term."%'";
                  }
            }
            if ($article_title!=""){
                  $syarat = $syarat." AND x.title LIKE '%
".$article_title." %'";
            }
            if ($journal_id!="" && $journal_id>0) {
                  $syarat = $syarat." AND
x.journal_id=".$journal_id;
            }
            if ($year!="") {
                  $syarat = $syarat." AND x.year=".$year;
            }
            if ($volume!="") {
                  $syarat = $syarat." AND x.volume=".$volume;
            }
            if ($issue!="") {
                  $syarat = $syarat." AND x.number=".$issue;
            }
            if ($author!="") {
                  $syarat = $syarat." AND (x.first_name LIKE
'%".$author."%' OR x.middle_name LIKE '%".$author."%' OR
x.last_name LIKE '%".$author."%')";
            }
            if ($issn!="") {
                  $sqlissn="SELECT journal_id from journal WHERE
issn='".$issn."' or eissn='".$issn."'";
                  $res=mysqli_query($this->db-
>con_tambahan,$sqlissn);

                  if ($dataissn=mysqli_fetch_assoc($res)) {
                        $syarat = $syarat." AND
x.journal_id=".$dataissn['journal_id'];}
                  else {
                        $syarat=$syarat." AND x.doi='".$issn."' ";
                  }
            }

            $sql="SELECT x.*
                  FROM (
                  SELECT a.article_id, a.journal_id, a.pages,
```

Segmen Program 4.7. *Class Article* (sambungan)

```
                  (SELECT setting_value FROM article_settings aset
WHERE setting_name='abstract' AND aset.article_id=a.article_id) as
abstract,
                  (SELECT setting_value FROM article_settings aset
WHERE setting_name='title' AND aset.article_id=a.article_id) as
title,
                  (SELECT setting_value FROM article_settings aset
WHERE setting_name='subject' AND aset.article_id=a.article_id) as
subject,
                  (SELECT setting_value FROM article_settings aset
WHERE setting_name='pub-id::doi' AND aset.article_id=a.article_id)
as doi,
                  (SELECT setting_value FROM journal_settings jset
WHERE setting_name='title' AND jset.journal_id=a.journal_id) as
journal_title,
                  (SELECT GROUP_CONCAT(first_name) FROM authors
auth WHERE auth.submission_id=a.article_id ORDER BY seq) as
first_name,
                  (SELECT GROUP_CONCAT(middle_name) FROM authors
auth WHERE auth.submission_id=a.article_id ORDER BY seq) as
middle_name,
                  (SELECT GROUP_CONCAT(last_name) FROM authors
auth WHERE auth.submission_id=a.article_id ORDER BY seq) as
last_name,
                  (SELECT year FROM issues isu WHERE
isu.issue_id=(SELECT issue_id FROM published_articles pa WHERE
pa.article_id=a.article_id)) as year,
                  (SELECT volume FROM issues isu WHERE
isu.issue_id=(SELECT issue_id FROM published_articles pa WHERE
pa.article_id=a.article_id)) as volume,
                  (SELECT number FROM issues isu WHERE
isu.issue_id=(SELECT issue_id FROM published_articles pa WHERE
pa.article_id=a.article_id)) as number
                  FROM articles a
                  ) x WHERE true ".$syarat;
            $res=mysqli_query($this->db->con,$sql);
            $return=array();
            while ($row=mysqli_fetch_assoc($res)) {
                  $return[]=$row;
            }
            return $return;
      }

      public function getAllArticleSubject(){
            $sql="SELECT a.article_id, aset.setting_value as
subject FROM articles a INNER JOIN (SELECT
article_id,setting_value FROM article_settings WHERE
setting_name='subject') as aset ON
(aset.article_id=a.article_id)";
```

Segmen Program 4.7. *Class Article* (sambungan)

```
	$res=mysqli_query($this->db->con,$sql);
		$return=array();
		while ($row=mysqli_fetch_assoc($res)) {
			$return[]=$row;
		}
		return $return;
	}
?>
```

4.5.3. *Class Author*

Class author berfungsi untuk mengambil data di basis data tentang penulis tersebut berdasarkan author_idnya, seperti dapat dilihat pada Segmen Program 4.8.

Segmen Program 4.8. *Class Author*

```
	<?php

	class Author{
		public
$db,$author_id,$submission_id,$seq,$first_name,$middle_name,$last_
name,$suffix,$country,$email,$url,$affiliation,$biography;

		public function __construct($db) {
			$this->db=$db;
		}

		public function load($id) {
			$sql="SELECT
x.author_id,x.submission_id,x.seq,x.first_name,x.middle_name,x.las
t_name,x.suffix,x.country,x.email,x.url,bio.setting_value	as
biography,aff.setting_value as affiliation
			FROM authors x
			LEFT JOIN (SELECT author_id, setting_name,
setting_value FROM author_settings WHERE setting_name='biography')
as bio ON (x.author_id=bio.author_id)
			LEFT JOIN (SELECT author_id, setting_name,
setting_value		FROM		author_settings		WHERE
setting_name='affiliation') as aff ON (x.author_id=aff.author_id)
			WHERE x.author_id=".$id;
			$res=mysqli_query($this->db->con,$sql);
			$data=mysqli_fetch_assoc($res);
			$this->author_id=$data['author_id'];
			$this->submission_id=$data['submission_id'];
			$this->seq=$data['seq'];
			$this->first_name=$data['first_name'];
			$this->middle_name=$data['middle_name'];
```

Segmen Program 4.8. *Class Author* (sambungan)

```
                $this->last_name=$data['last_name'];
                $this->suffix=$data['suffix'];
                $this->country=$data['country'];
                $this->email=$data['email'];
                $this->url=$data['url'];
                $this->affiliation=$data['affiliation'];
                $this->biography=$data['biography'];
        }

        public function getData() {
                return array(
                        'author_id'=>$this->author_id,
                        'submission_id'=>$this->submission_id,
                        'seq'=>$this->seq,
                        'first_name'=>$this->first_name,
                        'middle_name'=>$this->middle_name,
                        'last_name'=>$this->last_name,
                        'suffix'=>$this->suffix,
                        'country'=>$this->country,
                        'email'=>$this->email,
                        'url'=>$this->url,
                        'affiliation'=>$this->affiliation,
                        'biography'=>$this->biography
                );
        }
};
?>
```

4.5.4. *Class Journal*

Untuk melakukan load data-data judul jurnal sebagai pilihan di *advanced search*, maka dibuatlah *class journal* dengan fungsi getAllJournalTitle() seperti pada Segmen Program 4.9.

Segmen Program 4.9. C*lass Journal*

```
<?php
class Journal{
      public $db;

      public function __construct($db) {
            $this->db=$db;
      }
      public function getAllJournalTitle() {
            $sql="SELECT journal_id,setting_value FROM
`journal_settings` WHERE setting_name='title'";
            $res=mysqli_query($this->db->con,$sql);
            $return=array();
            while($row=mysqli_fetch_assoc($res)) {
```

Segmen Program 4.9. C*lass Journal* (sambungan)

```
	$return[]=$row;
		}
		return $return;
	}
};
?>
```

4.5.5. *Class Term*

Class term berfungsi untuk menyimpan data *indexing* di basis data tentang kata apa saja yang ada, dan jumlah *document frequency*, seperti dapat dilihat pada Segmen Program 4.10.

Segmen Program 4.10. *Class Term*

```
<?php
class Term{
	public $db, $term_id, $term_word,
$document_frequency,$maxID;

	public function __construct($db){
		$this->db=$db;
	}

	public function load($id){
		$sql="SELECT * FROM term WHERE term_id='".$id."';";
		$res=mysqli_query($this->db->con_tambahan,$sql);
		$data=mysqli_fetch_assoc($res);
		$this->term_id=$data['term_id'];
		$this->term_word=$data['term_word'];
		$this->document_frequency=$data['document_frequency'];
	}

	public function load_all() {
		$sql = "SELECT * FROM term";
		$res = mysqli_query($this->db->con_tambahan, $sql);
		$return = array();
		while ($row = mysqli_fetch_assoc($res)) {
			$return[] = $row;
		}
		return $return;
	}

	public function getData() {
		return array(
			'term_id'=>$this->term_id,
			'term_word'=>$this->term_word,
			'document_frequency'=>$this->document_frequency
```

Segmen Program 4.10. *Class Term* (sambungan)

```
            );
      }

      public function searchByTermWord($term) {
            $sql="SELECT * FROM term WHERE term_word='".$term."'";
            $res=mysqli_query($this->db->con_tambahan,$sql);
            $return=array();
            while ($row=mysqli_fetch_assoc($res)) {
                  $return[]=$row;
            }
            return $return;
      }

      public function deleteAllDataTerm() {
            $sql="DELETE FROM term WHERE 1";
            $res=mysqli_query($this->db->con_tambahan,$sql);
            if ($res) {
                  return array('status'=>1, 'msg'=>'Deleted all
term data');
            } else {
                  return array('status'=>0, 'msg'=>'Cannot
delete');
            }
      }
};
?>
```

4.5.6. *Class Tf*

Class tf berfungsi untuk menyimpan data *indexing* di basis data tentang kata terdapat di article_id mana, dan jumlah *term frequency* di article_id tersebut, seperti dapat dilihat pada Segmen Program 4.11.

Segmen Program 4.11. *Class Tf*

```
<?php
class Tf {
      public $db, $tf_id, $article_id, $term_id, $weight_tf;

      public function __construct($db) {
            $this->db=$db;
      }

      public function load ($id){
            $sql="SELECT * FROM tf WHERE tf_id='".$id."';";
            $res=mysqli_query($this->db->con_tambahan,$sql);
            $data=mysqli_fetch_assoc($res);
            $this->tf_id=$data['tf_id'];
            $this->article_id=$data['article_id'];
```

Segmen Program 4.11. *Class Tf* (sambungan)

```
      $this->term_id=$data['term_id'];
            $this->weight_tf=$data['weight_tf'];
      }

      public function load_all() {
            $sql = "SELECT * FROM tf";
            $res = mysqli_query($this->db->con_tambahan, $sql);
            $return = array();
            while ($row = mysqli_fetch_assoc($res)) {
                  $return[] = $row;
            }
            return $return;
      }

      public function getData() {
            return array(
                  'tf_id'=>$this->tf_id,
                  'article_id'=>$this->article_id,
                  'term_id'=>$this->term_id,
                  'weight_tf'=>$this->weight_tf
            );
      }

      public function searchByTermID($termID) {
            $sql="SELECT * FROM tf WHERE term_id='".$termID."'";
            $res=mysqli_query($this->db->con_tambahan,$sql);
            $return=array();
            while ($row=mysqli_fetch_assoc($res)) {
                  $return[]=$row;
            }
            return $return;
      }

      public function deleteAllDataTf() {
            $sql="DELETE FROM tf WHERE 1";
            $res=mysqli_query($this->db->con_tambahan,$sql);
            if ($res) {
                  return  array('status'=>1,  'msg'=>'Deleted  to
tf');
            } else {
                  return     array('status'=>0,     'msg'=>'Cannot
delete');
            }
      }
};
?>
```

4.5.7. Pemanggilan Fungsi-Fungsi pada *Class*

Untuk memanggil fungsi di *class*, dibutuhkan *file* tersendiri yang mengorganisir pemanggilan tiap fungsi. Pada Segmen Program 4.12. dapat dilihat salah satu cara pemanggilan fungsi, yaitu getAllArticleAbstract() di *class article*.

Segmen Program 4.12. Contoh Pemanggilan Fungsi pada *Class*

```
<?php
use \Psr\Http\Message\ServerRequestInterface as Request;
use \Psr\Http\Message\ResponseInterface as Response;

require '../vendor/autoload.php';
require 'database.php';
require 'models/article.php';
require 'models/term.php';
require 'models/tf.php';
require 'models/journal.php';
require 'models/author.php';
require 'models/irregular.php';

$db = new Database();
$app = new \Slim\App;
$app->get('/article',function(Request $request, Response
$response, array $args) {
    global $db;
    $article_model=new Article($db);
    $body=$article_model->getAllArticleAbstract();
    $response->getBody()->write(json_encode($body));
    $newResponse=$response->withHeader('Content-
type','application/json')->withHeader('Access-Control-Allow-
Origin','*');
    return $newResponse;
});
$app->run();

?>
```