PROGRAM APLIKASI

“IndoISDR: REST Web Services Instagram Spam Detection For Indonesian Language”

Dibuat dan dikembangkan oleh:

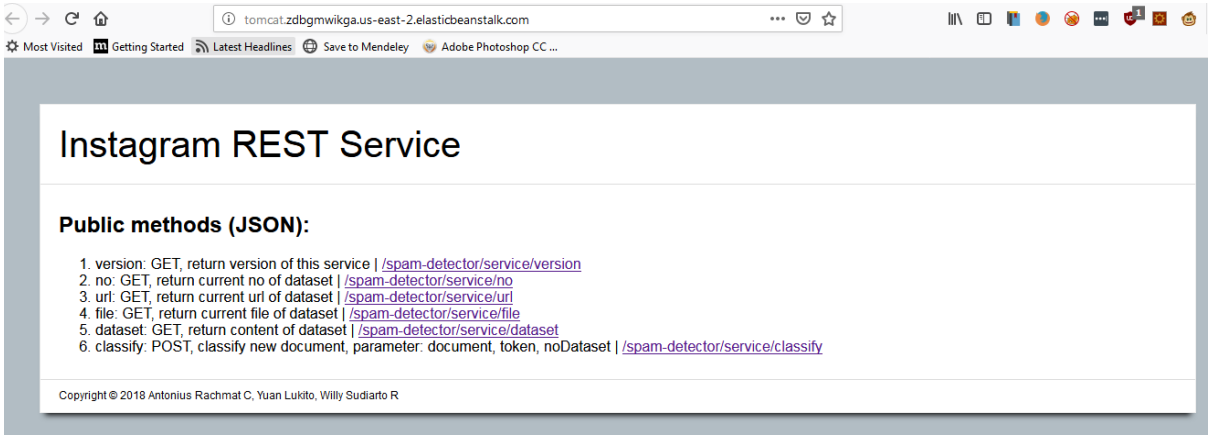
Antonius Rachmat Chrismanto, S.Kom., M.Cs.

Yuan Lukito, S.Kom., M.Cs.

Willy Sudiarto Raharjo, S.Kom., M.Cs.

Source Code Lengkap dapat didownload di
https://ti.ukdw.ac.id/~anton/publik/index.php?dir=&file=spam-detector.zip

Aplikasi dapat diakses di http://tomcat.zdbgmwikga.us-east-2.elasticbeanstalk.com/



Institusi:

Universitas Kristen Duta Wacana, Yogyakarta

Tahun: **2018**

SOURCE CODE APLIKASI

ClassifierRequest.java

```
package id.ac.ukdw.ti.research.instagram;

import javax.xml.bind.annotation.XmlRootElement;

@XmlRootElement
public class ClassifierRequest {
	private String document;
	private String token;
	private int noDataset;
	public String getDocument() {
		return document;
	}
	public void setDocument(String document) {
		this.document = document;
	}
	public String getToken() {
		return token;
	}
	public void setToken(String token) {
		this.token = token;
	}
	public int getNoDataset() {
		return noDataset;
	}
	public void setNoDataset(int noDataset) {
		this.noDataset = noDataset;
	}
}
```

IGClassifier.java

```
package id.ac.ukdw.ti.research.instagram;

import java.io.BufferedReader;
import java.io.FileInputStream;
import java.io.FileReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.URL;
import java.util.HashMap;
import java.util.Map;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.nio.file.StandardCopyOption;

import javax.json.*;
import com.opencsv.CSVReader;

import net.sf.javaml.core.kdtree.KDTree;

public class IGSpamClassifier {

	//API Version
	private static final String version = "2018-03-15.0.1";
```

```
	//Singleton object
	private static IGSpamClassifier classifier = new IGSpamClassifier();

	//Request tracking number
	private volatile long requestId = 1;

	//default message
	private static final String INVALID_TOKEN_MESSAGE = "Invalid token or
authentication code";

	//K-Nearest Neighbors
	private static final int K_PARAMETER = 5;
	private KDTree kdtree;
	private int noDataset = 1;

	//CSV path
	public String CSV_FILE_PATH = "https://s3.us-east-2.amazonaws.com/ig-
spam-detector-ukdw/komentar16000x1000unbal-nonstem.csv";
	public String LOCAL_FILE_PATH = "/tmp/dataset1.csv";
	private Map<String, Integer> tokenIndex = new HashMap<String,
Integer>();
	private int tokenCount;

	private IGSpamClassifier() {
		//load csv ke memory
		loadCSV();
	}

	private void loadCSV() {
		try {

	if(System.getProperty("os.name").toLowerCase().startsWith("windows"))
				LOCAL_FILE_PATH =
"tmp/dataset"+this.noDataset+".csv";

				Path target = Paths.get(LOCAL_FILE_PATH);
			if(!Files.exists(target)) {
				try (InputStream inp = new
URL(CSV_FILE_PATH).openStream()) {
					Files.copy(inp, target,
StandardCopyOption.REPLACE_EXISTING);
				}
			} else System.out.println("File sudah ada!!");

			System.out.println("CSV path: " +
target.toAbsolutePath().toString());

	System.out.println(System.getProperty("os.name").toLowerCase());

			FileInputStream fis = new
FileInputStream(LOCAL_FILE_PATH);
			CSVReader csvReader = new CSVReader(new
InputStreamReader(fis,"UTF-8"));
			String firstLine[] = csvReader.readNext();

			//susun token index
			for(int i=1; i<firstLine.length-1; i++) {
				tokenIndex.put(firstLine[i], i-1);
			}

			//inisialisasi kdtree
```

```
                int dimension = firstLine.length - 2;
                this.tokenCount = dimension;
                kdtree = new KDTree(dimension);

                //baca baris-baris berikutnya
                String[] nextRecord;
                while((nextRecord = csvReader.readNext()) != null) {
                      String classLabel = nextRecord[nextRecord.length-
1];
                      double[] tfidf = new double[dimension];
                      int featureIndex = 0;
                      for(int j=1; j<nextRecord.length-1; j++) {
                            tfidf[featureIndex] =
Double.parseDouble(nextRecord[j].trim());
                            featureIndex++;
                      }
                      kdtree.insert(tfidf, classLabel);
                }
                csvReader.close();
          }
          catch(IOException ioex) {
                ioex.printStackTrace();
          }
          catch(Exception ex) {
                ex.printStackTrace();
          }
    }

    public JsonObject getNoDataset() {
          JsonObject json = Json.createObjectBuilder()
                  .add("noDataset", this.noDataset)
                  .build();
                      return json;
    }

    public JsonObject getCSV_FILE_PATH() {
          JsonObject json = Json.createObjectBuilder()
                  .add("url", this.CSV_FILE_PATH)
                  .build();
                      return json;
    }

    public JsonObject getLOCAL_FILE_PATH() {
          JsonObject json = Json.createObjectBuilder()
                  .add("file", this.LOCAL_FILE_PATH)
                  .build();
                      return json;
    }

    public String getDataset() throws IOException {

    if(System.getProperty("os.name").toLowerCase().startsWith("windows"))
                LOCAL_FILE_PATH = "tmp/dataset"+this.noDataset+".csv";

          Path target = Paths.get(LOCAL_FILE_PATH);
          if(!Files.exists(target)) {
                try (InputStream inp = new
URL(CSV_FILE_PATH).openStream()) {
                      Files.copy(inp, target,
StandardCopyOption.REPLACE_EXISTING);
                }
```

```
            } else System.out.println("File sudah ada!!");
            System.out.println("CSV path: " +
target.toAbsolutePath().toString());

      System.out.println(System.getProperty("os.name").toLowerCase());

            BufferedReader br = new BufferedReader(new
FileReader(LOCAL_FILE_PATH));
            String line = br.readLine();
            StringBuilder sb = new StringBuilder();
          while (line != null) {
              sb.append(line);
              sb.append(System.lineSeparator());
              line = br.readLine();
          }
          String everything = sb.toString();
          br.close();
            return everything;
      }

      public static IGSpamClassifier getInstance() {
            return classifier;
      }

      public JsonObject getVersion() {
            JsonObject json = Json.createObjectBuilder()
        .add("version", version)
        .add("copyright", new java.text.SimpleDateFormat("yyyy").format(new
java.util.Date()))
        .add("creator", "antoniusrc,yuanlukito,willysr")
        .build();
            return json;
      }

      private boolean validateToken(String token) {
            return token.equalsIgnoreCase("yuanlukito");
      }

      private JsonObject invalidTokenJson() {
            JsonObject json = Json.createObjectBuilder().add("reason",
INVALID_TOKEN_MESSAGE).build();
            return json;
      }

      private String isSpam(String document) {
            //tokenisasi dokumen
            String[] token = document.split(" ");
            double[] features = new double[this.tokenCount];
            for(int i=0; i<features.length; i++)
                  features[i] = 0;
            for(String tok : token) {
                  System.out.println("Token: " + tok);
                  if(tokenIndex.containsKey(tok.toLowerCase())) {
                        int index = tokenIndex.get(tok.toLowerCase());
                        features[index] = 1;
                  }
            }
            Object o = kdtree.nearest(features);

            String labelResult = o.toString();
            System.out.println("Classification result: " + labelResult);
```

```
            return labelResult;
      }

      public JsonObject classifyDocument(String document, String token, int
noDataset) {
            //validasi token
            if(validateToken(token)) {
                  requestId++;
                  long requestStart = System.currentTimeMillis();
                  String classificationResult = isSpam(document);
                  long requestEnd = System.currentTimeMillis();

                  switch(noDataset) {
                  case 1:
                        if(this.noDataset!=noDataset) {
                              this.noDataset=noDataset;
                              this.CSV_FILE_PATH = "https://s3.us-east-
2.amazonaws.com/ig-spam-detector-ukdw/komentar16000x1000unbal-nonstem.csv";
                              this.LOCAL_FILE_PATH = "/tmp/dataset1.csv";

                              loadCSV();
                        }
                        break;
                  case 2:
                        if(this.noDataset!=noDataset) {
                              this.noDataset=noDataset;
                              this.CSV_FILE_PATH = "https://s3.us-east-
2.amazonaws.com/ig-spam-detector-ukdw/komentar16000x1000unbal-stem.csv";
                              this.LOCAL_FILE_PATH = "/tmp/dataset2.csv";

                              loadCSV();
                        }
                        break;
                  case 3:
                        if(this.noDataset!=noDataset) {
                              this.noDataset=noDataset;
                              this.CSV_FILE_PATH = "https://s3.us-east-
2.amazonaws.com/ig-spam-detector-ukdw/komentar16000x1000bal-nonstem.csv";
                              this.LOCAL_FILE_PATH = "/tmp/dataset3.csv";

                              loadCSV();
                        }
                        break;
                  case 4:
                        if(this.noDataset!=noDataset) {
                              this.noDataset=noDataset;
                              this.CSV_FILE_PATH = "https://s3.us-east-
2.amazonaws.com/ig-spam-detector-ukdw/komentar16000x1000bal-stem.csv";
                              this.LOCAL_FILE_PATH = "/tmp/dataset4.csv";

                              loadCSV();
                        }
                        break;
                  case 5:
                        if(this.noDataset!=noDataset) {
                              this.noDataset=noDataset;
                              this.CSV_FILE_PATH = "https://s3.us-east-
2.amazonaws.com/ig-spam-detector-ukdw/rapidminer-unbalanced-
nonstem16000x1000.csv";
```

```
                        this.LOCAL_FILE_PATH = "/tmp/dataset5.csv";

                        loadCSV();
                    }
                    break;
                case 6:
                    if(this.noDataset!=noDataset) {
                        this.noDataset=noDataset;
                        this.CSV_FILE_PATH = "https://s3.us-east-
2.amazonaws.com/ig-spam-detector-ukdw/rapidminer-unbalanced-
nonstem16000x1000.csv";
                        this.LOCAL_FILE_PATH = "/tmp/dataset6.csv";

                        loadCSV();
                    }
                    break;
                case 7:
                    if(this.noDataset!=noDataset) {
                        this.noDataset=noDataset;
                        this.CSV_FILE_PATH = "https://s3.us-east-
2.amazonaws.com/ig-spam-detector-ukdw/rapidminer-balanced-
nonstem16000x1000.csv";
                        this.LOCAL_FILE_PATH = "/tmp/dataset7.csv";

                        loadCSV();
                    }
                    break;
                case 8:
                    if(this.noDataset!=noDataset) {
                        this.noDataset=noDataset;
                        this.CSV_FILE_PATH = "https://s3.us-east-
2.amazonaws.com/ig-spam-detector-ukdw/rapidminer-balanced-
stem16000x1000.csv";
                        this.LOCAL_FILE_PATH = "/tmp/dataset8.csv";

                        loadCSV();
                    }
                    break;
                default:
                    this.noDataset=1;
                    this.CSV_FILE_PATH = "https://s3.us-east-
2.amazonaws.com/ig-spam-detector-ukdw/komentar16000x1000unbal-nonstem.csv";
                    this.LOCAL_FILE_PATH = "/tmp/dataset1.csv";

                    loadCSV();
                }

                JsonObject jsonRequestIdentifier =
Json.createObjectBuilder()
                        .add("requestid", requestId)
                        .add("start", requestStart)
                        .add("end", requestEnd)
                        .build();
                JsonObject json = Json.createObjectBuilder()
                        .add("request", jsonRequestIdentifier)
                        .add("document", document)
                        .add("noDocument", noDataset)
                        .add("result", classificationResult)
                        .build();
                return json;
            }
```

```
            else {
                  return invalidTokenJson();
            }
      }
}
```

IGSpamRESTServices.java

```
package id.ac.ukdw.ti.research.instagram;

import java.io.IOException;
import javax.ws.rs.Consumes;
import javax.ws.rs.GET;
import javax.ws.rs.POST;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;

@Path("/")
public class IGSpamRESTServices {
      private IGSpamClassifier classifier = IGSpamClassifier.getInstance();

      @GET
      @Produces(MediaType.APPLICATION_JSON)
      @Path("version")
      public String version() {
            return classifier.getVersion().toString();
      }

      @GET
      @Produces(MediaType.APPLICATION_JSON)
      @Path("no")
      public String no() {
            return classifier.getNoDataset().toString();
      }

      @GET
      @Produces(MediaType.APPLICATION_JSON)
      @Path("url")
      public String url() {
            return classifier.getCSV_FILE_PATH().toString();
      }

      @GET
      @Produces(MediaType.APPLICATION_JSON)
      @Path("file")
      public String file() {
            return classifier.getLOCAL_FILE_PATH().toString();
      }

      @GET
      @Produces(MediaType.TEXT_PLAIN)
      @Path("dataset")
      public String getDataset() throws IOException{
            return classifier.getDataset();
      }

      @POST
      @Consumes(MediaType.APPLICATION_JSON)
      @Produces(MediaType.APPLICATION_JSON)
      @Path("classify")
```

```
	public String classify(ClassifierRequest cRequest) {
		String document = cRequest.getDocument();
		String token = cRequest.getToken();
		int noDataset = cRequest.getNoDataset();
		return classifier.classifyDocument(document, token,
noDataset).toString();
	}
}
```

MANUAL PENGGUNAAN

"IndoISDR: REST Web Service's Instagram Spam Detection"

Antonius Rachmat Chrismanto
Yuan Lukito
Willy Sudiarto Raharjo

November 2018

Data Revisi Aplikasi

No. Release	Tanggal	Deskripsi
Rev. 0	2018-03-01.0.1	Initial Release
Rev. 1	2018-03-10.0.2	Revision Methods
Rev. 2	2018-03-15.0.3	Revision Dataset and Methods

Manual Penggunaan
Memo Otorisasi

Saya/Kami telah memeriksa seluruh dokumen Manual Penggunaan for IndoISDR: REST Web Service's Instagram Spam Detection. Dokumen ini telah memenuhi syarat dan kebutuhan menurut System Development Methodology.

SERTIFIKASI DOKUMEN – Silahkan pilih status dokumen.

__V__ Dokumen ini diterima.

_____ Dokumen ini masih pending dan perubahan masih dilakukan.

_____ Dokumen belum diterima.

Kami menerima perubahan untuk peningkatan dan otorisasi pekerjaan. Berdasarkan otoritas, keputusan, dan keberlanjutan kegiatan, sistem ini terotorisasi.

Antonius Rachmat Chrismanto — 16 November 2018
NAMA — TANGGAL
Project Leader

Yuan Lukito — 16 November 2018
NAMA — TANGGAL
Project Member

Willy Sudiarto Raharjo — 16 November 2018
NAMA — TANGGAL
Project Member

MANUAL PENGGUNAAN

DAFTAR ISI

HALAMAN #

1.0 INFORMASI UMUM

A. INFORMASI UMUM

1.1 Spesifikasi Sistem

Sistem ini telah dibangun dengan spesifikasi berikut:

- Software ini berbasis web menggunakan browser Firefox, Chrome, Safari, dan browser umum lainnya.
- REST web services.
- Instgram spam detection services.
- Judul: IndoISDR: REST Web Service Instagram Spam Detection.
- Kode Project: IGWS.
- Kategori software:
  - *Major application:* menyediakan layanan untuk deteksi komentar spam pada Instagram
  - *General support system:* digunakan pada browsers extension yang mengkonsumsi layanan web service
- Status Operasional:
  - Release v.3.0
  - Under development

1.2 Referensi Aplikasi

Referensi yang digunakan pada aplikasi ini adalah:

- JERSEY : https://jersey.github.io/
- ECLIPSE: https://www.eclipse.org/
- KD-TREE : http://java-ml.sourceforge.net/api/0.1.0/net/sf/javaml/core/kdtree/KDTree.html
- INSTAGRAM : https://www.instagram.com/
- AMAZON WEB SERVICE : http://console.aws.amazon.com/
- ECLIPSE AWS : https://www.eclipse.org/downloads/download.php?file=/oomph/epp/oxygen/R2/eclipse-inst-win64.exe
- KD-TREE: https://home.wlu.edu/~levys/software/kd/
- TOMCAT : http://tomcat.apache.org/download-80.cgi

1.3 Hak Akses Penggunaan

Kami menyediakan akses terhadap aplikasi pada web ("http://tomcat.zdbgmwikga.us-east-2.elasticbeanstalk.com/"), termasuk dokumentasi pada ("https://ti.ukdw.ac.id/~anton/publik/index.php?dir=&file=manualws.pdf") dan informasi produk terkait, contoh kode program, dan Application Program Interface information ("APIs"). Dokumentasi, layanan, dan API (termasuk jika ada update, perbaikan, fitur baru, atau penambahan properti.

1.4 Kontak

1.4.1 Informasi

Untuk informasi dapat menghubungi:

Antonius Rachmat Chrismanto, S.Kom., M.Cs.
Jl. Dr. Wahidin Sudiro Husodo No. 5-25 Yogyakarta
Email: anton@ti.ukdw.ac.id

1.5 Manual Penggunaan

Manual Penggunaan v0.01.

2.0 PENJELASAN SISTEM

B. PENJELASAN SISTEM

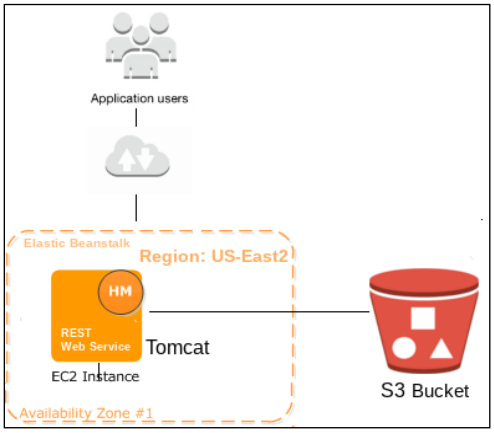
2.1 Konfigurasi Sistem

Sistem ini dibangun di atas Amazon Web Services (AWS) pada alamat http://console.aws.amazon.com. Sistem Amazon Web Services (AWS) adalah platform layanan cloud aman, yang menawarkan daya komputasi, penyimpanan database, pengiriman konten, dan fungsionalitas lain untuk membantu menskalakan dan mengembangkan bisnis. Data kebutuhan sistem dapat dilihat pada Tabel 1.

Tabel 1. Tabel Layanan AWS yang Digunakan

No.	Kebutuhan	Layanan AWS	URL	Keterangan
1.	Tempat penyimpanan dataset CSV berjumlah 8 versi dengan ukuran total mencapai 30mb x 8 = 240 mb	Amazon S3	https://s3.console.aws.amazon.com/	Gratis, Free Tier
2.	Server untuk menyimpan sekaligus deploy web service berbasis Java / NET	Amazon EC2	https://ec2.console.aws.amazon.com	Gratis, Free Tier
3.	Aplikasi untuk Web Service	Amazon Elastics Beanstalk	https://us-east-2.console.aws.amazon.com/elasticbeanstalk	Gratis, Free Tier

Desain arsitektur AWS yang digunakan dapat dilihat pada gambar 1.



Gambar 1. Arsitektur AWS pada Sistem

2.2 Aliran Data

Aliran data pada sistem ini adalah:

1. Pengguna mengakses alamat web service di http://tomcat.zdbgmwikga.us-east-2.elasticbeanstalk.com/
2. Pengguna dapat mengkases method berikut:

- Fungsi classify: fungsi ini digunakan untuk mengklasifikasikan suatu dokumen / data baru dalam method POST. Input dari fungsi ini adalah dokumen yang akan diklasifikasikan, dan token.
- Fungsi version: fungsi ini digunakan untuk menampilkan versi web service yang dibuat dalam method GET. Fungsi ini tidak memiliki input.
- Fungsi url: fungsi ini digunakan untuk menampilkan URL dari dataset learning yang sedang digunakan sekarang dalam method GET. Fungsi ini tidak memiliki input.
- Fungsi file: fungsi ini digunakan untuk mengambil isi file dataset yang sedang aktif digunakan dalam format text dengan method GET. Fungsi ini tidak memiliki input
- Fungsi dataset: fungsi ini digunakan untuk mengganti nomor file dataset yang digunakan sebagai sumber data pelatihan sistem dalam method GET. Fungsi ini menggunakan input 1 buah data, yaitu nomor dataset 1 s/d 8.

3. Alamat yang dapat diakses:
    a. Fungsi version: http://tomcat.zdbgmwikga.us-east-2.elasticbeanstalk.com/service/version (GET)
    b. Fungsi no: http://tomcat.zdbgmwikga.us-east-2.elasticbeanstalk.com/service/no (GET)
    c. Fungsi url: http://tomcat.zdbgmwikga.us-east-2.elasticbeanstalk.com/service/url (GET)
    d. Fungsi file: http://tomcat.zdbgmwikga.us-east-2.elasticbeanstalk.com/service/file (GET)
    e. Fungsi dataset: http://tomcat.zdbgmwikga.us-east-2.elasticbeanstalk.com/service/dataset (GET)
    f. Fungsi classify: http://tomcat.zdbgmwikga.us-east-2.elasticbeanstalk.com/service/classify (POST)
4. Penggunaan dilakukan secara nyata pada level browsers extension.

2.3 Hak Akses Pengguna

Hak akses pengguna adalah sebagai berikut:

1. Pengguna yang dapat mengakses IndoISDR ini adalah developer aplikasi yang akan mengklasifikasikan / mendeteksi komentar spam pada Instagram
2. Pengguna harus menggunakan token rahasia yang dapat diminta kepada pengembang aplikasi pada bab 1 di atas.

3.0 PENGGUNAAN SISTEM

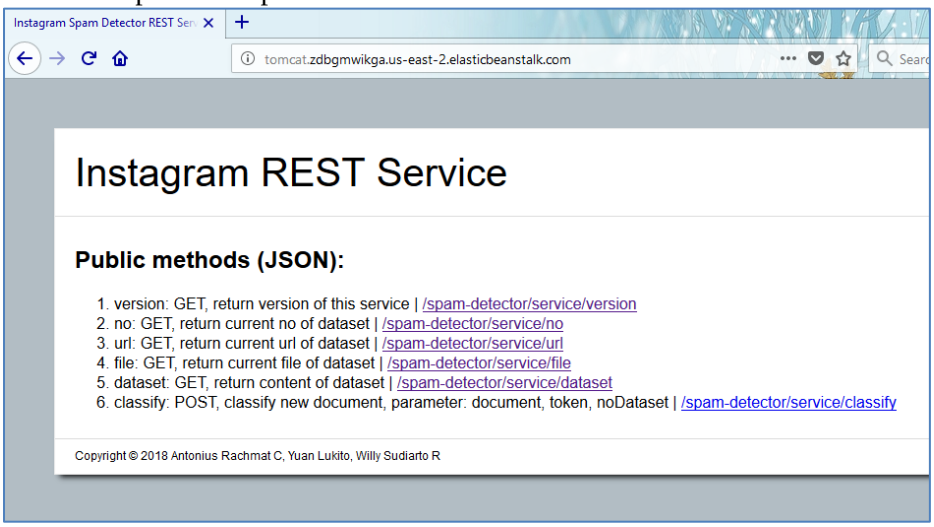
C. PENGGUNAAN SISTEM

3.1 Logging On

Untuk dapat mengakses IndoISDR dapat dilakukan dengan 2 cara:

1. Melalui website http://tomcat.zdbgmwikga.us-east-2.elasticbeanstalk.com/
2. Melalui browser extension yang mengakses http://tomcat.zdbgmwikga.us-east-2.elasticbeanstalk.com/

Tampilan awal sistem dapat dilihat pada Gambar 2.

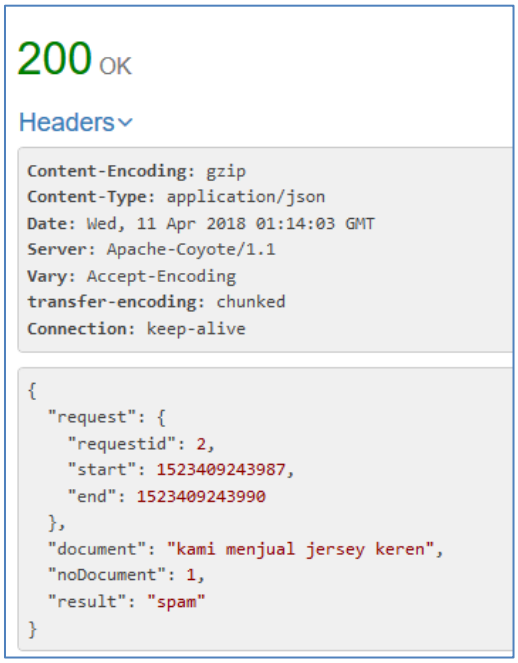


Gambar 2. Tampilan Awal Sistem

3.2 Penggunaan Method / Layanan

3.2.1 Method Classify

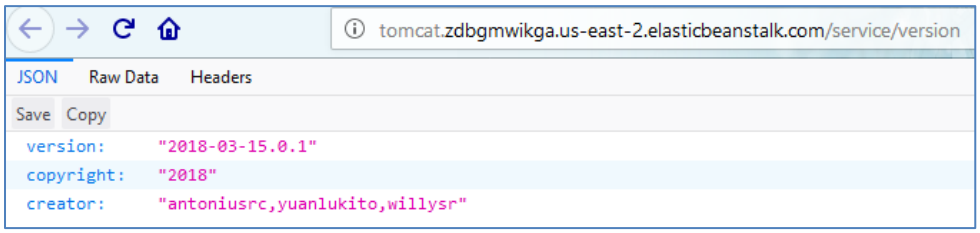
Fungsi ini digunakan untuk mengklasifikasikan suatu dokumen / data baru dalam method POST. Input dari fungsi ini adalah dokumen yang akan diklasifikasikan, dan token. Tampilan hasil pemanggilan fungsi Classify menggunakan method REST POST dengan parameter isi komentar, token, dan noDataset dapat dilihat pada Gambar 5.3. Fungsi juga sudah berhasil menerima dan memproses method POST dan mengeluarkan output JSON sesuai dengan yang diharapkan.



Gambar 3. Hasil dari Method Classify

3.2.2 Method Version

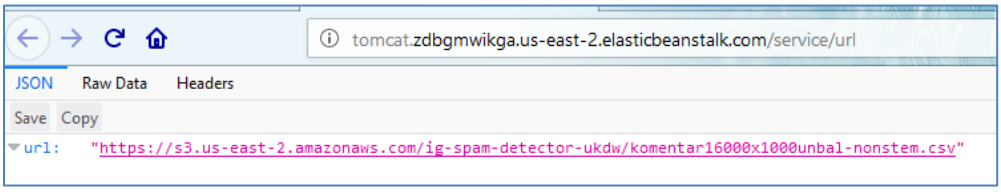
Fungsi ini digunakan untuk menampilkan versi web service yang dibuat dalam method GET. Fungsi ini tidak memiliki input. Hasil method version dapat dilihat di Gambar 4.



Gambar 4. Hasil dari Method Version

3.2.3 Method Url

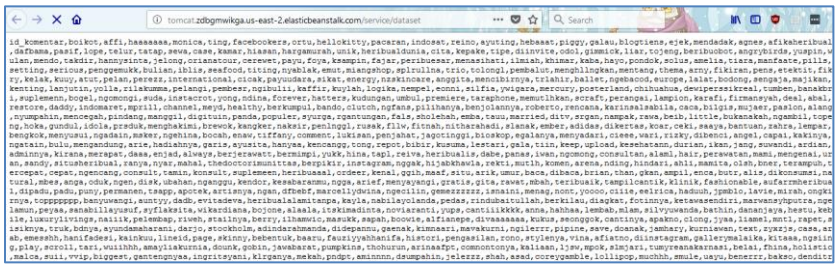
Fungsi ini digunakan untuk menampilkan URL dari dataset learning yang sedang digunakan sekarang dalam method GET. Fungsi ini tidak memiliki input. Hasil method version dapat dilihat di Gambar 5.



Gambar 5. Hasil dari Method URL

3.2.4 Method File

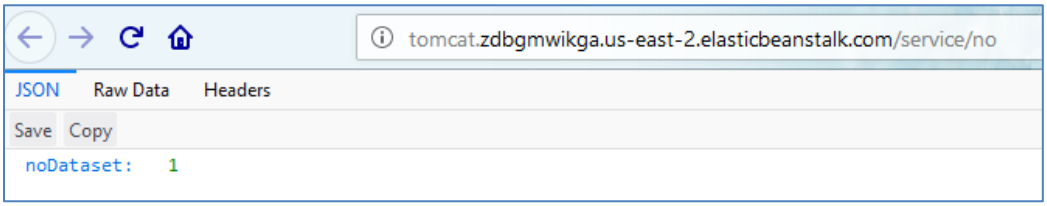
Fungsi ini digunakan untuk mengambil isi file dataset yang sedang aktif digunakan dalam format text dengan method GET. Fungsi ini tidak memiliki input. Hasil method version dapat dilihat di Gambar 6.



Gambar 6. Hasil dari Method File

3.2.5 Method Dataset

Fungsi ini digunakan untuk mengganti nomor file dataset yang digunakan sebagai sumber data pelatihan sistem dalam method GET. Fungsi ini menggunakan input 1 buah data, yaitu nomor dataset 1 s/d 8. Hasil method version dapat dilihat di Gambar 7.



Gambar 7. Hasil dari Method Dataset

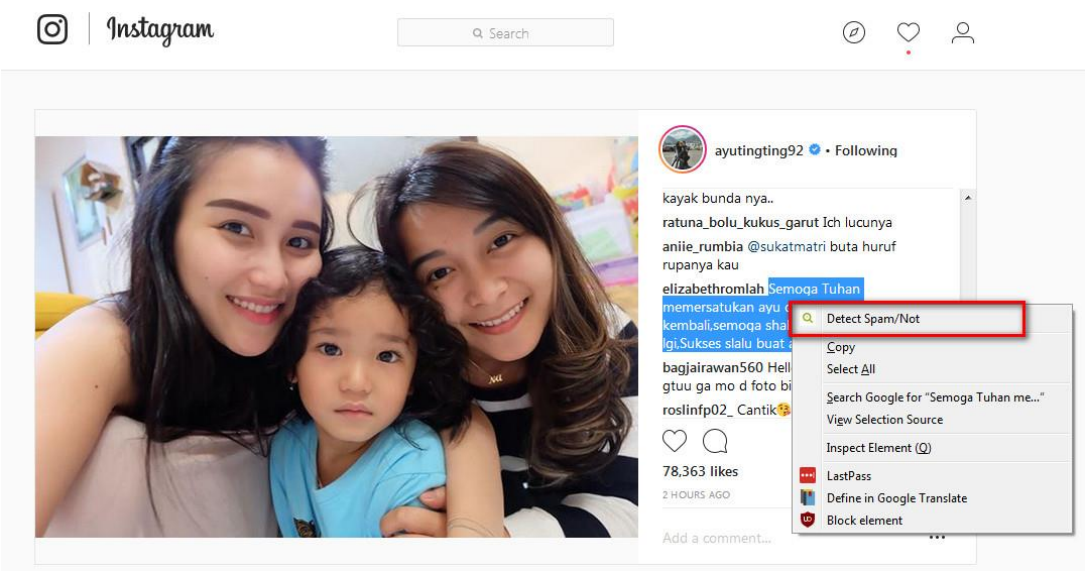
3.3 Penggunaan Aplikasi dari Firefox Extension

Prototipe browser extension diimplementasikan untuk browser Firefox. Browser extension dibangun di atas plugin GreaseMonkey yang dapat didownload di https://addons.mozilla.org/en-US/firefox/addon/greasemonkey/ untuk Firefox.

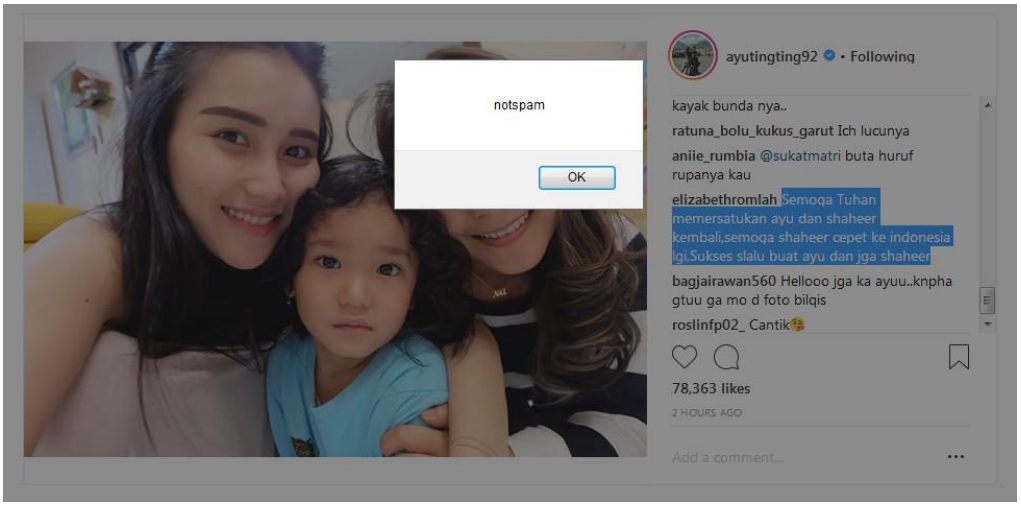
User script yang dibuat, dibuat dengan menggunakan Javascript yang akan memiliki cara kerja sebagai berikut:

- Script akan memeriksa apakah halaman yang dibuat adalah instagram? Jika instagram maka script akan membuat menu klik kanan pada firefox bernama Check Spam/Not
- Ketika user menseleksi suatu kalimat pada IG, kemudian mengklik Check Spam/Not sistem akan membaca teks yang terpilih dan mengirimkan ke web service sesuai dengan protokol yang berlaku.
- Sistem akan membaca hasil dari web service dan menampilkannya dalam bentuk message box di browser

Hasil dari prototipe plugin Firefox dapat dilihat pada gambar 8, 9 dan 10.



Gambar 8. Tampilan Plugin Firefox pada Instagram (1)



Gambar 9. Tampilan Plugin Firefox pada Instagram (2)

Gambar 10. Tampilan Plugin Firefox pada Instagram (3)

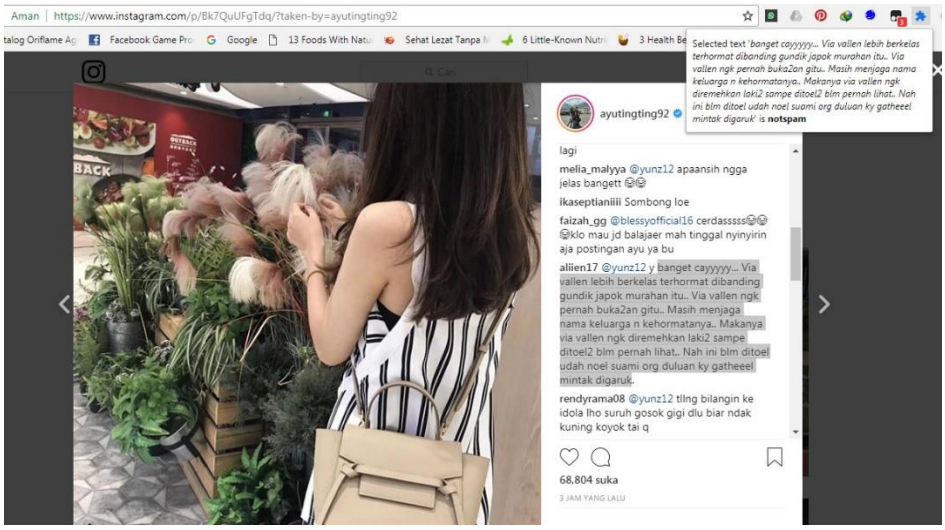
3.4 Penggunaan Aplikasi dari Chrome Extension

Prototipe browser extension diimplementasikan untuk browser Chrome. Browser extension dibangun di atas plugin GreaseMonkey yang dapat didownload di https://chrome.google.com/webstore/detail/tampermonkey/dhdgffkkebhmkfjojejmpbldmpobfkfo?hl=en untuk Chrome.

User script yang dibuat, dibuat dengan menggunakan Javascript yang akan memiliki cara kerja sebagai berikut:

- Script akan memeriksa apakah halaman yang dibuat adalah instagram? Jika instagram maka script akan membuat menu klik kanan pada firefox bernama Check Spam/Not
- Ketika user menseleksi suatu kalimat pada IG, kemudian mengklik Check Spam/Not sistem akan membaca teks yang terpilih dan mengirimkan ke web service sesuai dengan protokol yang berlaku.
- Sistem akan membaca hasil dari web service dan menampilkannya dalam bentuk message box di browser

Hasil dari prototipe plugin Firefox dapat dilihat pada gambar 11.



Gambar 11. Tampilan Plugin Chrome pada Instagram

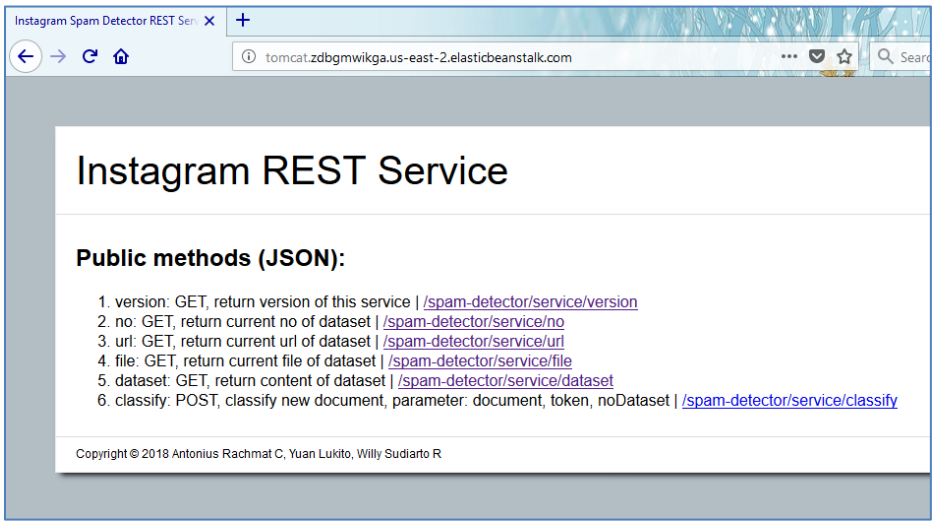
3.5 Exit System

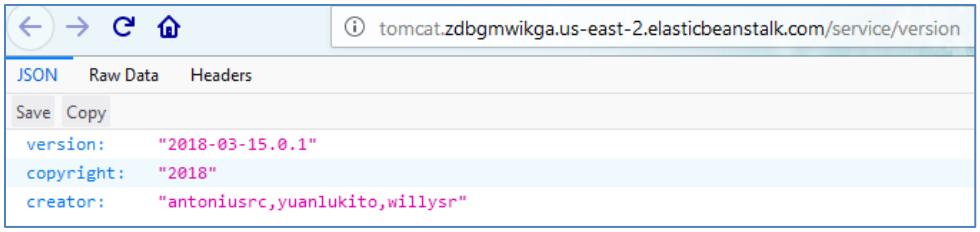
Untuk keluar dari sistem, silahkan tinggalkan / close website.

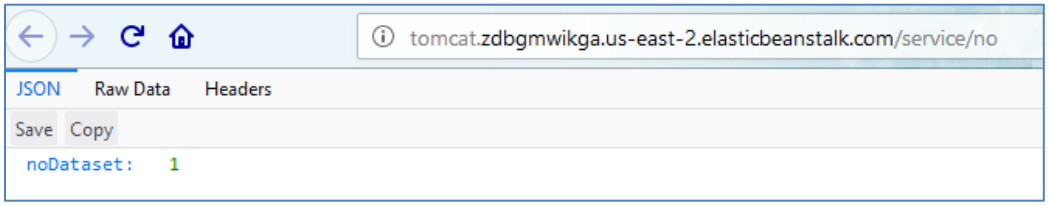
10.0 LAMPIRAN

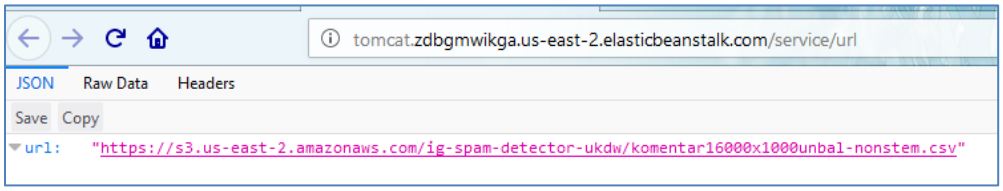
A. LAMPIRAN

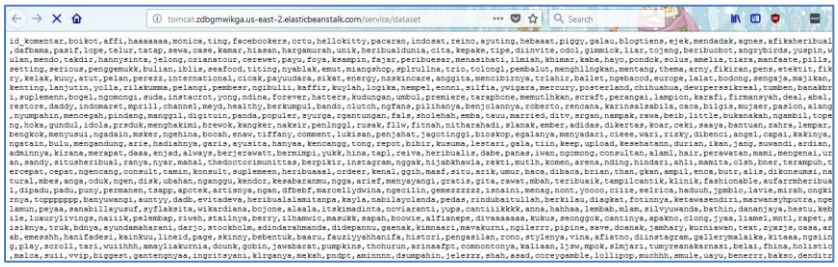
Tampilan Sistem:

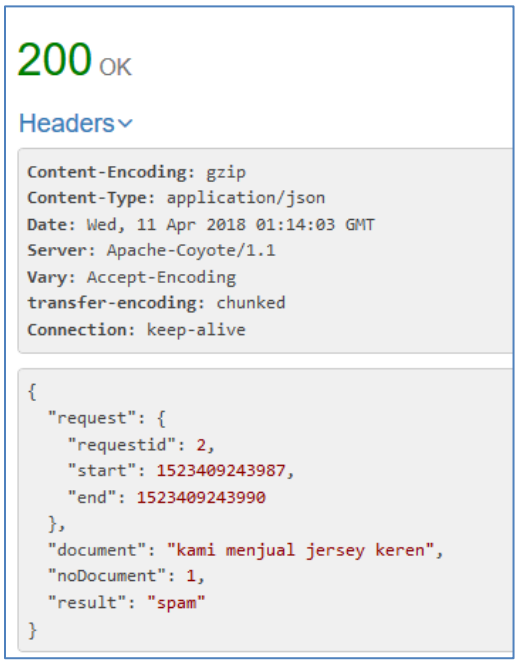












200 OK

Headers

```
Content-Encoding: gzip
Content-Type: application/json
Date: Wed, 11 Apr 2018 01:14:03 GMT
Server: Apache-Coyote/1.1
Vary: Accept-Encoding
transfer-encoding: chunked
Connection: keep-alive
```


```
{
  "request": {
    "requestid": 2,
    "start": 1523409243987,
    "end": 1523409243990
  },
  "document": "kami menjual jersey keren",
  "noDocument": 1,
  "result": "spam"
}
```