Penerapan Algoritma Pencocokan String dalam Review Highlights di Google Play Store

Vitra Chandra (13514043)
Program Studi Teknik Informatika
Institut Teknologi Bandung
Bandung, Indonesia
13514043@std.stei.itb.ac.id

***Abstract*—Google Play Store adalah suatu aplikasi layanan konten digital yang dikelola google. Pada masa kini, seluruh aplikasi yang diunduh pengguna dapat diulas oleh masing-masing pengguna untuk menjadi umpan balik bagi developer aplikasi tersebut. Akan tetapi, kebanyakan ulasan pengguna tenggelam dalam banyaknya ulasan sehingga gampang diabaikan. Google menerapkan *review highlights*, sebuah fitur pada aplikasi Google Play yang secara langsung memproses setiap ulasan dari pengguna dan memberikan suatu garis besar dari ulasan-ulasan tersebut. Makalah ini akan membahas pentingnya algoritma pencocokan string dan bagaimana cara *review highlights* bekerja.**

Keywords—google play store; string matching; review highlights;

I. PENDAHULUAN

Google Play Store adalah aplikasi resmi dari Google sebagai layanan pengunduhan aplikasi lainnya untuk Android. Pada Google Play Store, pengguna dapat memberikan ulasan terhadap aplikasi yang diunduhnya dalam bentuk rating 1 sampai 5 atau juga dengan tambahan komentar. Komentar yang diberikan pengguna berupa suatu teks yang berisi pujian maupun keluhan dari pengguna.

Aplikasi yang tersedia pada Google Play Store tidaklah sedikit. Susah bagi developer untuk selalu meninjau setiap aplikasi yang dikelolanya, terutama pada bagian ulasan. Suatu ulasan yang diberikan pengguna biasanya tidak terlalu signifikan. Akan tetapi, jika ulasan yang hampir mirip diberikan oleh banyak pengguna, makan ulasan tersebut sudah menggambarkan aplikasi tersebut. Dari banyaknya ulasan-ulasan yang diberikan pengguna, biasanya ada yang tenggelam dalam ulasan-ulasan lainya sehingga tidak terlihat dan mudah terabaikan.

Selain itu, bagian ulasan juga dapat digunakan oleh pengguna. Pengguna dapat membaca ulasan sebelum mengunduh suatu aplikasi untuk melihat bagaimana pandangan pengguna lain terhadap aplikasi tersebut. Akan tetapi, banyaknya ulasan juga menyusahkan pengguna untuk memperoleh informasi yang dia inginkan dan jarang ada pengguna yang memiliki cukup waktu untuk membaca setiap ulasan yang ada.

Dari pertimbangan diatas Google meluncurkan suatu fitur baru pada Google Play Store yaitu *review highlights.* Pada *review highlights,* pengguna dapat melihat secara garis besar apa yang pengguna lainya lihat dari aplikasi tersebut. Selain itu, *review highlights* juga memperlihatkan pengguna berapa banyak ulasan yang mengatakan hal tersebut. *Review highlights* juga berguna bagi developer karena developer tidak perlu meninjau setiap ulasan untuk melihat garis besarnya.

II. REVIEW HIGHLIGHTS

Review Highlights memproses setiap ulasan yang diberikan pengguna dan mengembalikan suatu kata yang sering dilihat pada bagian ulasan aplikasi tersebut. Fitur *review highlights* digunakan pada aplikasi yang ulasan penggunanya sudah sangat banyak sehingga perlu diberikan garis besarnya. Untuk aplikasi yang masih sedikit pengguna tidak digunakan fitur tersebut karena tidak bisa menentukan garis besarnya. Hal ini terlepas dari urutan peringkat aplikasi pada Google Play Store sehingga setiap aplikasi yang ulasannya sudah memenuhi suatu batas dimana dianggap banyak dapat menampilkan *review highlights* yang diambil dari ulasan pengguna.

Dibawah ini contoh *review highlights* pada aplikasi Clash Royale.

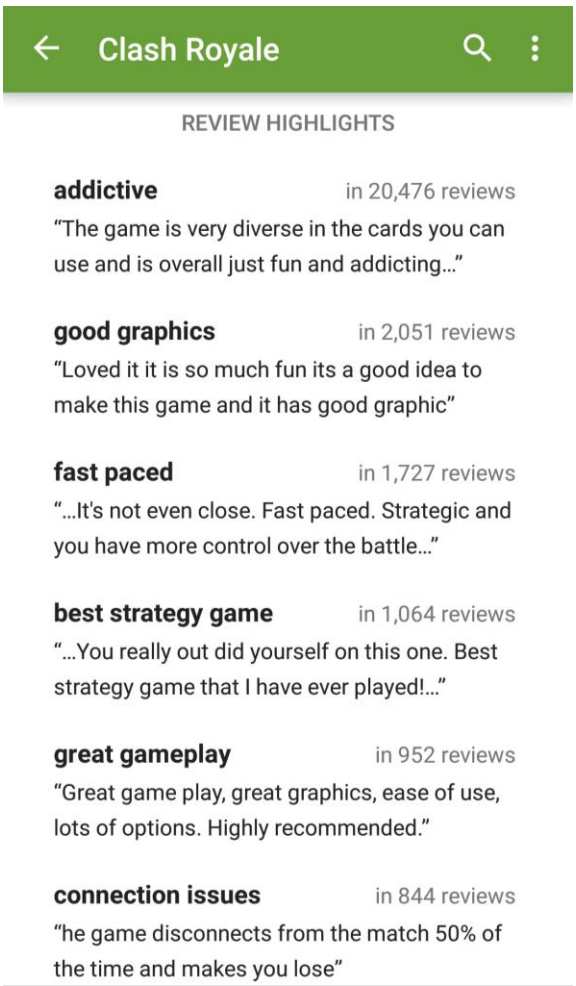

Fig. 1. Clash Royale review highlights.

Pada contoh diatas, *review highlights* memberikan suatu pendapat sama dari ulasna-ulasna pengguna. "addictive", "good graphics", "fast paced" merupakan gambaran besar yang dapat diambil dari ulasan-ulasan yang diberikan pengguna.

III. Dasar Teori

A. *Konsep String*

Asumsikan S suatu string dengan ukuran m, maka:

$$S = x_1x_2\ldots x_m$$

Suatu awalan (*prefix*) dari S adalah setiap upastring S yang memenuhi S[1…k-1] dan suatu akhiran (*suffix*) adalah setiap upastring S yang memenuhi S[k-1…m] dimana k adalah suatu indeks diantara 1 dan m dan S[0] adalah suatu karakter null dengan simbol ∅. Contoh:

S: "andrew"

Semua awalan yang mungkin adalah "∅", "a", "an", "and", "andr", "andre" dan semua akhiran yang mungkin adalah "∅", "w", "ew", "rew", "drew", "ndrew".

B. *Pencocokan String (String Matching)*

Algoritma pencocokan string atau pencarian string digunakan untuk mencari letak suatu string atau pattern pada suatu teks. Persoalan pencarian string dirumuskan sebagai berikut:

- Diberikan:
  - Teks (long string) dengan panjang n karakter
  - Pattern (string) dengan panjang m karakter (m < n) yang dicari dalam teks.
- Cari lokasi pertama pada teks dimana pattern sesuai dengan teks.

Persoalan diatas dapat digambarkan seperti ini:

Pattern: "hari"

Teks: "kami pulang ke rumah ketika **hari** mulai malam"

Pattern didalam teks yang cocok dengan pattern yang dicari disebut sebagai target.

Terdapat beberapa algoritma pencocokan string diantaranya:

1) *Algoritma Brute Force*

Algoritma Brute Force melakukan pencocokan pada setiap karakter di teks. Dengan asumsi bahwa teks berupa array [1…n] dan pattern berupa array [1…m], maka algoritma brute force pencocokan string adalah sebagai berikut:

a) Pattern P dicocokkan dengan awal teks

b) Bandingkan setiap karakter dalam pattern dengan karakter pada teks sampai ditemukansemua karakter pada pattern sama dengan bagian pada teks atau dijumpai ketidak cocokan karakter.

c) Sebelum teks habis, jika pattern tidak cocok, maka geser pattern ke kanan sebanyak 1 karakter dan ulangi langkah pada (b).

Contoh:

```
Pattern: NOT
Teks: NOBODY NOTICED HIM

  NOBODY NOTICED HIM
1 NO
2  N
3   N
4    N
5     N
6      N
7       N
8        NOT
```

Pada contoh diatas, pattern NOT ditemukan pada posisi indeks ke 8 dari teks.

Pseudocode algoritma brute force adalah sebagai berikut:

```
repeat
  if (text letter == pattern letter) then
    compare next letter of patter to next
letter of text
  else
    move pattern down text by one letter
until (entire pattern found or end of text)
```

Kompleksitas algoritma brute force dalam kasus terbaik adalah O(n) bila setiap karakter pada pattern tidak pernah sama dengan pada teks. Sedangkan, kasus terburuk membutuhkan m(n – m + 1) dimana kompleksitasnya O(mn).

2) Algoritma Knuth-Morris-Pratt (KMP)

Seperti halnya algoritma brute force, algoritma KMP melakukan pencarian dari kiri ke kanan. Akan tetapi, algoritma KMP menyimpan informasi pencarian yang telah dilakukannya, sehingga mencegah terjadinya perbandingan yang sia-sia. Algoritma KMP menggunakan fugsi pinggiran (*Border functio*) untuk menentukan panjang string yang perlu digesernya agar tidak terjadi pencocokan yang sia-sia. Fungsi pinggiran bergantung pada karakter dalam pattern dan bukan pada teks sehingga fungsi pinggiran dihitung sebelum dilakukan pencarian.

Fungsi pinggiran b() didefinisikan sebagai panjang awalan dari pattern yang juga merupakan akhiran dari pattern.

Contoh:

Pattern: ababaa

j	1	2	3	4	5	6
P[i]	a	b	a	c	a	b
b(j)	0	0	1	0	1	0

Fungsi b() mengembalikan nilai yang menyatakan banyaknya pergeseran yang dapat dihindari (dilewati).

Contoh pencarian menggunakan algoritma KMP dengan menggunakan fungsi pinggiran diatas:

```
Teks: abacaabaccabacabaabb
Pattern: abacab

  abacaabaccabacabaabb
1 abacab
2     ab
3       abaca
4           a
5            abacab
```

Pada contoh diatas, ketika pencocokan menyatakan tidak cocok pada pencocokan ke-6 (baris 1), fungsi pinggiran dipanggil dan pattern digeser sebanyak 5 - 1 karakter.

Pseudocode algoritma KMP adalah sebagai berikut:

```
Build border function b()
while (pattern not found and not end of text)
do
  if (pattern letter == text letter) then
    compare next letter of patter to next
letter of text
  else if (not first check) then
    next pattern index = b(current pattern
index - 1)
  else
    check next text letter
```

Komplexitas algoritma KMP dibagi 2 yaitu menghitung fungsi pinggiran dan pencarian string. Fungsi pinggiran memerlukan kompleksitas O(m) dan pencarian string memerlukan kompleksitas O(n), sehingga kompleksitas totalnya adalah O(m+n).

3) Algoritma Boyer-Moore

Algoritma Boyer-Moore merupakan variasi dari algoritma string matching dengan melompat sejauh mungkin. Berbeda dengan KMP dan brute force, algoritma Boyer-Moore melakukan perbandingan dari kanan ke kiri. Pencocokan stirng algoritma Boyer-Moore dilakukan dengan 2 teknik yaitu:

a) The looking-glass technique: cari pattern dalam text dari belakang text.

b) The character-jump technique: ketika ketidakcocokan terjadi, geser pattern pada text sebanyak suatu nilai untuk mencegah pencocokan yang sia-sia.

Algoritma Boyer-Moore menggunakan fungsi kemunculan terakhir (*Last Occurrence Function*) L() dimana pattern akan diproses bersama kumpulan karakter A terlebih dahulu sebelum dilakukan pencarian pattern pada text. L() memetakan setiap karakter dalam A sebagai integer. Fungsi L(x) dimana x bagian dari A didefinisikan sebagai indeks terbesar dimana pattern == x, atau -1 jika tidak ada indeks tersebut. Biasanya L() disimpan dalam suatu array.

Contoh:

A: {a, b, c, d}
Pattern: abacab

x	a	b	c	d
L(x)	5	6	4	-1

Fungsi L() mengembalikan indeks kemunculan terakhir dari tiap karakter pada pattern di A.

Contoh pencarian menggunakan algoritma Boyer-Moore dengan fungsi kemunculan terakhir diatas:

```
Teks: abacaabadcabacabaabb
Pattern: abacab

  abacaabadcabacabaabb
1      b
2     cab
```

```
3         b
4          b
5                b
6            abacab
```

Pada contoh diatas, ketika pencocokan menyatakan tidak cocok, fungsi L() dipanggil dan pattern digeser sebanyak panjang pattern - L(x) dengan x merupakan huruf pada text yang tidak cocok.

Pseudocode algoritma Boyer-Moore adalah sebagai berikut:

```
Build last occurrence function L()
repeat
  if (pattern letter == text letter) then
    if (current pattern index == 0) then
      return current text index
    else
      check previous letter
  else
    jump pattern according to L() function
until (pattern found or end of text)
```

Kompleksitas algoritma Boyer-Moore sebesar O(nm + A) karena perhitungan yang sama dengan algoritma brute force. Akan tetapi, pada penggunaannya algoritma Boyer-Moore jauh lebih cepat dari algoritma brute force dalam pencarian teks bacaan.

C. *Approximate String Matching*

Approximate string matching adalah suatu penurunan dari algoritma pencocokan string untuk mencari pattern pada string dengan suatu batas kesalahan. Batas kesalahan yang digunakan dalam *approximate string matching* disebut sebagai *edit distance*. *Approximate string matching* digunakan untuk mencari sugesti kata atau kesalahan cetak/ketik pada suatu teks. Pada *approximate string matching*, terdapat 3 operasi primitif yaitu:

1) *insertion:* blan → b**u**lan
2) *deletion:* b**u**lan → blan
3) *substitution:* bu**l**an → bu**k**an

Ketiga operasi diatas dapat dilakukan dengan mensubstitusikan karakter yang akan dimasukkan dan dihapus dengan karakter null ∅.

Algoritma *approximate string matching* terbagi atas 2 kategori, yaitu: *on-line* dan *off-line. On-line* melakukan pencarian dengan memproses pattern tanpa menggunakan index.

D. *Natural Language Processing*

Natural Language Processing (NLP) merupakan suatu cabang ilmu pada bidang studi informatika, intelejensi buatan, dan komputasi linguistik. NLP memyangkut pembelajaran interaksi manusia dengan komputer. NLP banyak digunakan untuk memperoleh data dari masukkan manusia.

NLP menggunakan *machine learning* untuk membuat komputer memiliki kemampuan untuk mengerti bahasa manusia. Pengaplikasian ini dilakukan dengan menggunakan *approximate string matching* karena mungkin saja suatu input dari manusia memiliki kesalahan. *Machine learning* memungkinkan komputer untuk mengerti tidak hanya kata tetapi juga frasa sehingga suatu kalimat dapat dipecah dan diambil intinya.

IV. Analisis Permasalahan

Review highlights pada Google Play Store mencari kata-kata pada ulasan-ulasan yang diberikan pengguna untuk dijadikan suatu gambaran besar dari aplikasi tersebut. Hal ini dapat dilakukan dengan mencari kata-kata yang dapat menggambarkan suatu aplikasi, seperti: menarik, grafik, koneksi, indah, dsb. Kata-kata tersebut dimasukkan kedalam suatu kamus untuk kemudian dilakukan pencariannya.

Kata-kata dalam kamus tersebut tidak hanya dalam bentuk satu kata tetapi dapat dalam bentuk frasa. Selain itu, kata-kata tersebut juga bisa dalam *part of speech* yang berbeda, seperti: *addicting*, *addictive*, dan *addicted*.

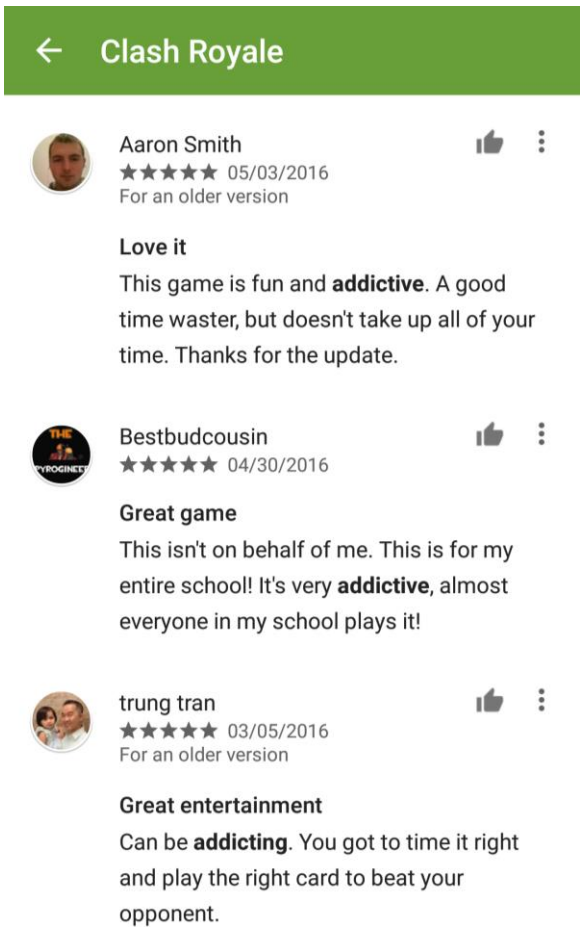

Fig. 2. Clash Royale review highlights addictive category

Pencarian kata-kata diatas menggunakan *approximate string searching* sehingga dapat diperoleh bahwa *addictive* dan addicting adalah suatu kata yang mengandung arti sama. Hal ini perlu menggunakan NLP karena sebelum ditemukannya bahwa kata *addictive* banyak ditemukan di ulasan, kata *addictive* juga harus berupa suatu kata yang valid dan berarti sehingga dapat dijadikan *review highlights.*

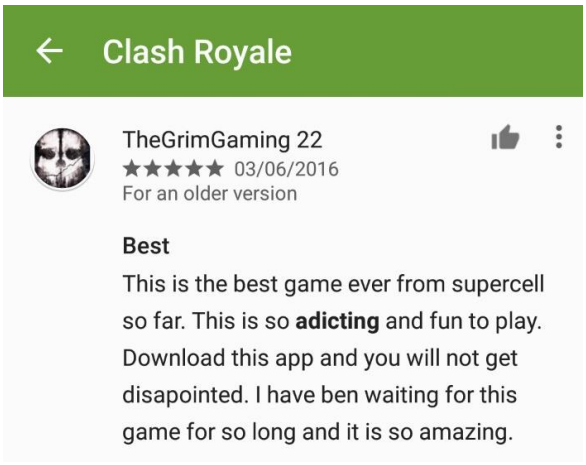

Fig. 3. Clash Royale review highlights addictive category with typo

Gambar diatas menggambarkan bahwa pencarian juga dapat terjadi walaupun ulasan pengguna mengandung kesalahan ketikan.

Pencarian juga dilakukan pada frasa yang sering digunakan seperti: *good graphics*, *fast paced*, *best strategy game*. Pencarian pada frasa juga dapat menerima kesalahan ketikan atau perbedaan kata-kata, seperti: *great gameplay*, dan *great game play*. Berikut contoh dari pencarian tersebut:

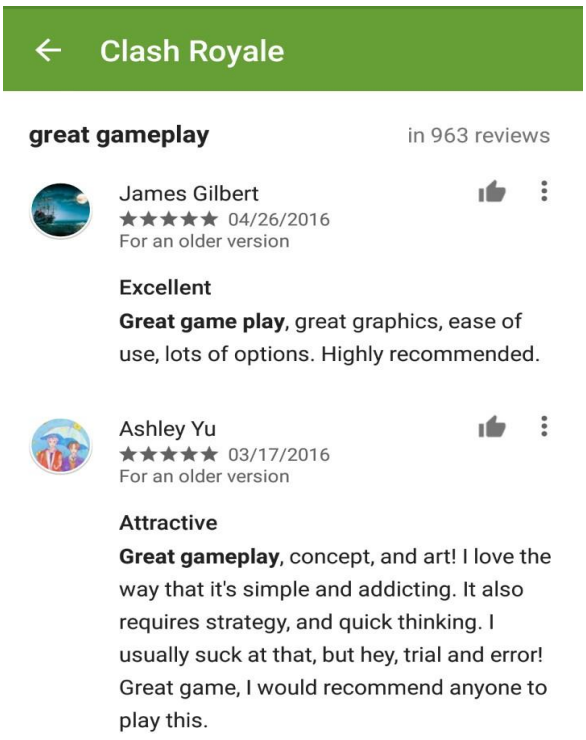

Fig. 4. Clash Royale review great gameplay category

Pencarian kata-kata juga tidak dilakukan secara sembarangan karena jika diambil bahwa kata-kata yang paling sering bermunculan, makan kata *I, game, of* akan banyak bermunculan.

Selain itu, algoritma *string matching* digunakan kedua kalinya ketika pengguna menekan *highlight* yang dipilih seperti *addictive.* Ketika pengguna menekan *highlight* tersebut, program akan melakukan *approximate string matching* untuk mencari kata-kata yang mirip *highlight.* Setelah pencarian dilakukan, Google Play Store menampilkan ulasan-ulasan yang merupakan bagian dari *highlight* tersebut.

V. Analisis String Matching

Algoritma string matching yang digunakan dalam pengaplikasian *review highlights* pada Google Play Store untuk mencari kata-kata pada ulasan-ulasan yang diberikan pengguna tidak mungkin menggunakan algoritma brute force. Hal ini dikarenakan algoritma brute force melakukan pencarian dengan mencocokkan setiap karakter dan akan memakan sangat banyak waktu, terutama bila ulasan yang tersedia sudah dalam hitungan puluhan ribu sampai ratusan ribu.

Algoritma KMP dan algoritma Boyer-Moore lebih cocok dalam hal ini karena penggunaannya yang jauh lebih cepat daripada algoritma brute force. Akan tetapi, algoritma KMP ataupun algoritma Boyer-Moore belumlah optimal sehingga perlu dioptimasi. Mungkin saja pada kenyataanya, Google tidak menggunakan kedua algoritma ini dalam merealisasikan *review highlights*.

VI. Kesimpulan

Review highlights pada Google Play Store merupakan slaah satu contoh penerapan *high-level language processing* yang memerlukan NLP. NLP memerlukan *machine learning* untuk menentukan kata-kata yang perlu dipilih dan dijadikan *highlight. Approximate string matching* merupakan bagian penting dari pencarian tersebut karena kesalahan yang minimal perlu diabaikan. *Approximate string matching* dapat diimplementasikan dengan algoritma *string matching* sehingga dapat diperoleh bahwa algoritma dasar seperti *string matching* sangatlah penting. Tanpa adanya algoritma tersebut, maka *review highlights* dari Google Play Store tidak dapat direalsasikan.

Ucapan Terima Kasih

Penulis mengucapkan terima kasih kepada Tuhan Yang Maha Esa atas bimbingannya sehingga makalah ini dapat terselesaikan secara tepat waktu, kepada Orang Tua penulis atas perhatian, cinta kasih, dan didikan yang diberikan sehingga dia tumbuh dan berkembang sebagai seorang pelajar yang baik dan dapat menjadi mahasiswa Teknik Informatika Institut Teknologi Bandung, dan kepada Bapak Rinaldi Munir dan Ibu Nur Ulfa Maulidevi atas ajaran Strategi Algoritma sehingga makalah ini dapat terselesaikan dengan baik.

REFERENSI

[1] R. Munir, *Strategi Algoritma*, Bandung: Informatika, 2003.

[2] *Strings and Pattern Matching.* Diakses pada May 4, 2016. Dari: https://www.cs.purdue.edu/homes/ayg/CS251/slides/chap11.pdf

[3] R. Baeza-Yates, G. Navarro. *Fast Approximate String Matching in a Ditionary*. University of Chile: Dept. of Computer Science.

[4] *About Goolge Play*. Google. Diakses pada May 4, 2016. Dari: https://support.google.com/googleplay/

[5] *Google Play Store*. Android Central. Diakses pada May 4, 2016. Dari: http://www.androidcentral.com/google-play-store/

[6] *Google Play now lets Android app developer see review highlihts, ratings charts.* Venture Beat. Diakses pada May 5, 2016. Dari: http://venturebeat.com/2016/02/25/google-play-now-lets-android-app-developers-see-review-highlights-ratings-charts/

[7] *Google Experiments With Review Highlights In The Play Store.* Android Police. Diakses pada May 5, 2016. Dari: http://www.androidpolice.com/2015/12/15/google-experiments-with-review-highlights-in-the-play-store/

[8] *New tools for ratings & reviews on Google Play to engage and understand your users.* F. Hurley, Android Developers Blog. Diakses pada May 6, 2016. Dari: http://android-developers.blogspot.co.id/2016/02/new-tools-for-ratings-reviews-on-google.html

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 10 Desember 2015

Vitra Chandra
13514043