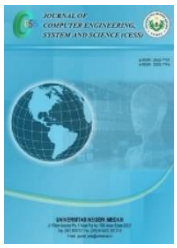
CESS
(Journal of Computing Engineering, System and Science) 8(2) July 2023 483-494
e-ISSN: 2502-714x
p-ISSN: 2502-7131

Contents list available at www.jurnal.unimed.ac.id

CESS
(Journal of Computing Engineering, System and Science)

journal homepage: https://jurnal.unimed.ac.id/2012/index.php/cess



Kolaborasi Algoritma Stemming Nazief & Adriani Dengan Metode Parsing Queries PostgreSQL Untuk Pencarian Nama Program Studi Baru Vokasi

Collaboration of Nazief & Adriani Stemming Algorithm with PostgreSQL Queries Parsing Method to Search for New Study Program Names

Indra Chaidir

Program Studi Sistem Informasi Universitas Bina Sarana Informatika
Jl. Kramat Raya No. 98, Senen. Jakarta Pusat, Indonesia.
email: indra@bsi.ac.id

A B S T R A K

Penolakan usulan nama baru program studi vokasi pada Aplikasi Silemkerma di Direktorat Jenderal Pendidikan Tinggi Vokasi, Kementerian Pendidikan, Kebudayaan, Riset, dan Teknologi sering terjadi karena terdapat kemiripan nama program studi yang diusulkan dengan nama program studi yang sudah ada di dalam basis data. Banyak data tidak ditemukan karena filter data menggunakan metode konvensional dalam kasus ini menggunakan operator ILIKE dengan pola *wildcard character* % (percent), sedangkan data yang dicari tersedia di dalam basis data. Ini terjadi dikarenakan operator ILIKE tidak dapat membaca perubahan kata dari leksem/akar kata (root word) seperti "**pengelolaan**" dengan memiliki prefix dan suffix, dengan akar kata "**kelola**". Mengatasi permasalahan ini, penulis memanfaatkan Algoritma Nazief & Adriani untuk stemming agar mendapatkan leksem dari kalimat yang dimasukan. Hasil algoritma tersebut terus diolah menggunakan Metode Parsing Queries, salah satu metode Full Text Search yang ada pada basis data PostgresQL. Hasil penelitian ini dapat diimplementasikan pada Aplikasi tersebut.

Kata Kunci: *Natural Language Processing, Stemming, Parsing Queries Full Text Search, PostgreSQL, Program Studi Baru, Nomenklatur*

A B S T R A C T

Rejection of new vocational study program name proposals in Silemkerma Application at the Directorate General of Vocational Higher Education, Ministry of Education, Culture, Research, and Technology often occurs because there is a similarity between the proposed study program name and the existing study program name in the database. Many data are not found because the data filter uses conventional methods in this case using the ILIKE operator with the wildcard character pattern % (percent), while the data sought is available in the database.

**Penulis Korespondensi:*
email: indra@bsi.ac.id

This is because the ILIKE operator cannot read word changes from lexemes/root words such as "**pengelolaan**" which has a prefix and suffix, with the root word "**kelola**". Overcoming this problem, the author utilizes the Nazief & Adriani Algorithm for stemming in order to get lexemes from the sentences entered. The results of the algorithm are then processed using the Parsing Queries Method, one of the Full Text Search methods available in the PostgresQL database. The results of this research can be implemented in the application.

Keywords: *Natural Language Processing, Stemming, Parsing Queries, Full Text Search, PostgreSQL, New Study Programs, Nomenclature*

1. PENDAHULUAN

Menjalankan suatu aplikasi oleh pengguna sudah sesuai dengan pedoman atau buku manual. Disisi sistem, informasi yang dimasukan oleh pengguna juga sudah benar. Tetapi informasi dimasukan tidak sesuai baik disisi pengguna maupun disisi sistem seperti pencarian data yang tidak ditemukan dalam basis data. Padahal informasi yang dicari sebenarnya sudah ada didalam basis data ataupun memiliki kemiripan informasi yang dimasukkan sudah ada didalam basis data.

Pada layanan Sistem Informasi Pengembangan Kelembagaan Perguruan Tinggi atau disingkat SILEMKERMA, terdapat fitur usulan nomenklatur program studi baru atau disebut juga penamaan program studi baru. Sering terjadi usulan penamaan program studi baru ditolak, karena nama program studi tersebut sudah ada pada basis data atau memiliki kemiripan nama.

Metode sebelumnya menggunakan ILIKE sebagai operator filter untuk query data menggunakan teknik pencocokan pola. Hasilnya berupa string case-insensitive dan menampilan hasil sesuai dengan pola yang disebutkan.

```
SELECT nama_prodi, kdjenjang FROM tbl_prodi WHERE nama_prodi ilike '%informatika%';
```

Gambar 1. Query filter menggunakan operator ILIKE

Pada gambar 1 menerangkan metode pencarian data menggunakan operator ILIKE dengan pola : wildcard character % (percent) dan kata yang dicari , contoh : '% informatika%' . hasilnya akan menampilkan nama program studi yang memiliki kata informatika.

Tabel 1. Hasil Query ILIKE

Pendidikan Komputer atau Informatika
Ilmu Komputer/Informatika
Pendidikan Vokasional Informatika (Ilmu Komputer)
Bio-informatika

Tapi beberapa kasus tidak dapat menemukan nama program studi sedangkan di basis data memiliki nama program studi tersebut, contoh query:

```
select nama_prodi
  from tbl_prodi
        where nama_prodi ilike '%pengolahan hasil ikan%';
```

Query diatas tidak menemukan nama program studi pengolahan hasil ikan, sedangkan di basis data terdapat beberapa nama program studi yang memiliki kemiripan. Sistem menganggap program studi tersebut belum ada dan perguruan tinggi dapat mengusulkan nomenklatur baru tersebut. Tim penilai akan menolak usulan tersebut karena sudah ada nama program studi yang menyerupai nama program studi yang diusulkan. Permasalahan karena perubahan kata hingga perbedaan nama.

Terkait permasalahan ini, penulis menggunakan metode Parsing Queries yang merupakan salah satu metode *full text search* yang terdapat pada basis data PostgreSQL. Sebelum menggunakan Parsing Queries, kalimat atau kata yang dimasukan, terlebih dahulu di-stemming menggunakan algoritma stemming bahasa Indonesia dan Algoritma *Stemming* yang digunakan adalah Algoritma Nazief dan Andriani. Algoritma Stemming menggunakan pemroses bahasa alami atau biasa dikenal dengan *Natural Language Processing (NLP)*. NLP penting karena membantu dalam menyelesaikan ambiguitas dalam bahasa dan menambahkan struktur pada data untuk analisis teks dan pengenalan suara[1] .

2. TINJAUAN TEORI

Pada penelitian ini melibatkan studi morfologi. Morfologi adalah ilmu bahasa mengenai seluk-beluk bentuk kata, bagaimana kata-kata tersebut itu terbentuk dan pengaruh perubahan kata seperti adanya awalan dan akhiran pada akar kata. Morfologi kata Bahasa Indonesia dapat terdiri dari struktur infleksi dan derivasional. Infleksi adalah struktur paling sederhana yang dinyatakan dengan sufiks yang tidak mempengaruhi makna dasar dari kata dasar. yang tidak mempengaruhi makna dasar dari kata dasar yang mendasarinya [2].

2.1. Stemming

Stemming adalah proses menghasilkan varian morfologi dari dasar kata (*root word*). Dengan kata sederhana, ini mengurangi suatu kata menjadi kata dasarnya[3]. Contoh stemming kata : kata "pengelolaan" didapatkan kata dasarnya "kelola". Stemming pada kata bahasa Indonesia berbeda dengan stemming pada kata dalam bahasa inggris. Pada bahasa inggris proses yang diperlukan adalah menghilangkan sufiks saja, sedangkan pada bahasa Indonesia selain menghilangkan sufiks juga menghilangkan prefix dan juga konfiks[2]. Algoritma Stemming Bahasa Indonesia pertama kali adalah Algoritma Porter. Kemudian algoritma ini diteliti dan dikembangkan oleh Nazief & Adriani[4]. Algoritma *stemming* Nazief & Adriani memiliki hasil yang lebih bagus dibandingan dengan algoritma *stemming* porter. Namun waktu proses, algoritma porter lebih baik dibandingkan dengan algoritma Nazief & Adriani [5].

2.2. Algoritma Nazief & Adriani

Algoritma stemming Nazief & Adriani sangat cocok di terapkan dalam proses pencarian kata dasar dalam Bahasa Indonesia berdasarkan kamus kata dasar[6]. Pada penelitian ini peneliti menggunakan Algoritma Nazief & Adriani untuk pencocokan kata dasar dengan infomasi yang diterima berupa teks. Algoritma ini didasarkan pada aturan morfologi Bahasa Indonesia yang luas, yang dikumpulkan menjadi satu kelompok dan dienkapsulasi menjadi imbuhan yang diizinkan dan imbuhan yang tidak diizinkan. Terdapat kamus kata dasar yang digunakan untuk mendukung perekaman kata dan pencocokan kata seletah dilakukan stemming kata [7]. Proses stemming pada algoritma Nazief & Adriani sebagai berikut:

1) Sebelum proses selanjutnya kata akan dicocokan pada database kata dasar. Apabila ada yang cocok maka algoritma berhenti.
2) Hapus akhiran (“-lah”, “-kah”, “-tah” atau “-pun”) jika itu adalah partikel (“-lah”, “-kah”, “-tah” atau “- pun”) langkah ini diulangi lagi untuk menghilangkaninflectional possessive pronoun suffixes (“-ku”, “-mu” atau “-nya”). Periksa kata dalam kamus kata dasar, jika ditemukan, algoritma berhenti.
3) Hapus Derivational Suffix (“-i” atau “-an”). Periksa kata dalam kamus kata dasar. Jika kata tersebut ditemukan dalam kamus kata dasar, maka algoritma berhenti. Jika tidak ditemukan maka proses lanjut ke langkah 3a:
   a. Jika ada akhiran “-an” dan huruf terakhir adalah “-k” maka “-k” akan dihapus. Periksa kata dalam kamus kata dasar. Jika kata tersebut ditemukan dalam kamus maka proses berhenti. Jika tidak, maka lanjut ke langkah 3b.
   b. Akhiran yang dihapus (“-i”, “-an”, atau “-kan”) tidak jadi dilakukan menghapusan imbuhan dan lanjut ke langkah 4.
4) Hapus Derivational Prefix (“be-”, “di-”, “ke-”, “me-”, “pe-”, “se-” dan “te-”). Periksa kata dalam kamus kata dasar. Jika kata tersebut ditemukan dalam database kata dasar, maka proses berhenti. Pada tahapan ini berhenti jika memenuhi kondisi berikut:
   a. Ditemukan kombinasi awalan dan akhiran yang tidak diizinkan.
   b. Awalan yang ditemukan sama dengan awalan yang dihapus sebelumnya.
   c. Menghilangkan tiga awalan
5) Jika proses sudah dilakukan tetapi kata dasar tidak ditemukan pada database kamus kata dasar, makakata asli yang belum dilakukan stemming akan dikembalikan.

Algoritma Nazief & Adriani telah memiliki pustaka (*library*) bahasa pemograman seperti bahasa pemograman Java, C, Python, Go, Ruby dan PHP. *Library* yang digunakan adalah pustaka Satrawi berbasis bahasa pemograman *Hypertext Preprocessor* (PHP), dapat dilihat pada laman http://sastrawi.github.io. Proses stemming oleh Sastrawi sangat bergantung pada kamus kata dasar. Sastrawi menggunakan kamus kata dasar dari kateglo.com dengan sedikit perubahan.

Library Sastrawi PHP ini menggunakan Algoritma Nazief & Adriani, kemudian menyesuaikan pengembangan algoritma dalam penelitian Asian[8] , Arifin[9] dan Tahitoe[10]. Asian dkk melakukan beberapa pengembangan algoritma Nazief & Adriani sebagai berikut:

1) Menggunakan kamus kata yang lebih lengkap
2) Menambahkan aturan-aturan untuk kata-kata majemuk perulangan.
3) Menambahkan aturan awalan dan akhira, serta aturan lainnya, yaitu:
   a. Menambahkan perikel (inflection suffix) “-pun”.
   b. Penambahan aturan pemenggalan awalan.
   c. Perubahan aturan pemnggalan untuk tipe awalan “me”.
4) Perubahan urutan proses stemming, yaitu:
   a. Kata dengan awalan “be-” dan akhiran “-lah”, hilangkan awal terlebih dahulu kemudian akhiran.
   b. Kata dengan awalan “be-” dan akhiran “-an”, hilangkan awalan terlebih dahulu kemudian akhiran.
   c. Kata dengan awalan “me-” dan akhiran “-i”, hilangkan awalan terlebih dahulu kemudian akhiran.

d. Kata dengan awalan “di-” dan akhiran “-i”, hilangkan awalan terlebih dahulu kemudian akhiran.
e. Kata dengan awalan “pe-” dan akhiran “-i”, hilangkan awalan terlebih dahulu kemudian akhiran.
f. Kata dengan awalan “ter-” dan akhiran “-i”, hilangkan awalan terlebih dahulu kemudian akhiran.

2.3. Full Text Search

Pada sistem basis data PostgreSQL, terdapat fitur *Full Text Search*, yaitu menyediakan kemampuan untuk mengidentifikasi dokumen berbahasa alami yang memenuhi kueri, dan secara opsional untuk mengurutkannya berdasarkan relevansi dengan kueri. Jenis pencarian yang paling umum adalah untuk menemukan semua dokumen yang mengandung istilah kueri yang diberikan dan mengembalikannya dalam urutan kemiripannya dengan kueri. Pengertian query dan kemiripan sangat fleksibel dan tergantung pada aplikasi tertentu. Pencarian yang paling sederhana menganggap query sebagai sekumpulan kata dan kemiripan sebagai frekuensi kata query dalam dokumen [11].

Untuk mengimplementasikan pencarian teks lengkap, harus ada fungsi untuk membuat tsvector dari dokumen dan tsquery dari kueri pengguna. Selain itu, kita perlu mengembalikan hasil dalam urutan yang berguna, jadi kita memerlukan fungsi yang membandingkan dokumen sehubungan dengan relevansinya dengan kueri. Juga penting untuk dapat menampilkan hasilnya dengan baik. PostgreSQL menyediakan dukungan untuk semua fungsi ini.

a. Parsing Document
b. Parsing Queries
c. Rangking Search Results
d. Highlighting Results

2.4. Parsing Queries

Metode ini merupakan salah satu metode full text search yang terdapat pada database PostgreSQL versi 11 keatas. Pada penelitian ini menggunakan metode Parsing Queries untuk menjalankan leksem kata yang sudah diolah terlebih dahulu oleh Algoritma Nazief & Adriani.

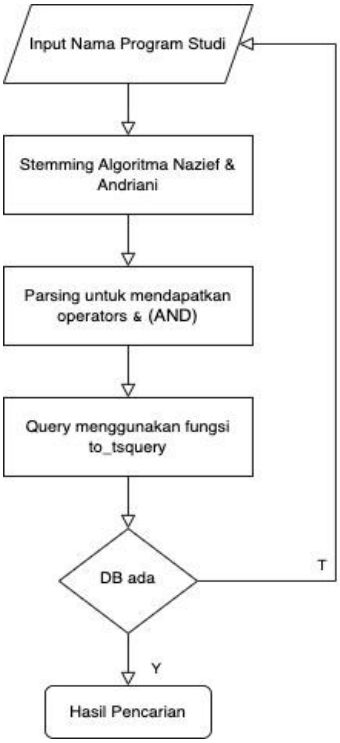
PostgreSQL menyediakan fungsi to_tsquery, plainto_tsquery, phraseto_tsquery, dan websearch_to_tsquery untuk mengubah kueri menjadi tipe data tsquery. to_tsquery menyediakan akses ke lebih banyak fitur dibandingkan plainto_tsquery atau phraseto_tsquery, tetapi kurang memaafkan masukannya. websearch_to_tsquery merupakan versi sederhana dari to_tsquery dengan sintaks alternatif, mirip dengan yang digunakan oleh mesin pencari web.

to_tsquery membuat nilai tsquery dari querytext, yang harus terdiri dari token-token tunggal yang dipisahkan oleh operator tsquery & (AND), | (OR), ! (NOT), dan <-> (FOLLOWED BY), yang dapat dikelompokkan dengan menggunakan tanda kurung. Dengan kata lain, input ke to_tsquery harus sudah mengikuti aturan umum untuk input tsquerya. Perbedaannya adalah bahwa ketika input tsquery dasar mengambil token-token secara mentah, to_tsquery menormalkan setiap token menjadi leksem menggunakan konfigurasi yang ditentukan atau default, dan membuang setiap token yang merupakan stop words sesuai dengan konfigurasi.

Sebagai contoh:
SELECT to_tsquery(Kelola & Kebun & Kopi);

3. METODE

Pada penelitian ini, untuk mendapatkan nama program studi yang tepat dan atau memiliki kemiripan dengan nama program studi yang sudah ada dalam basis data, maka peneliti melakukan alur sebagai berikut:

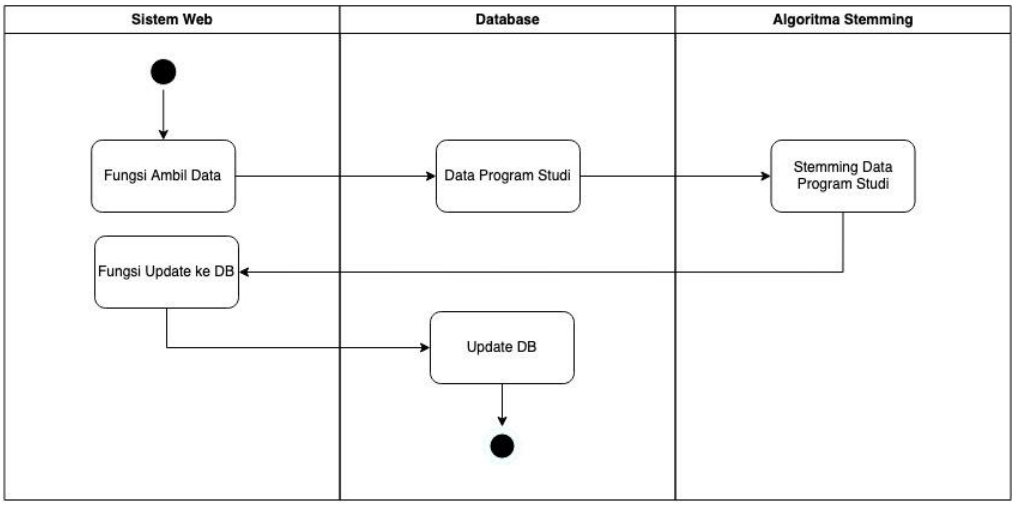


Gambar 2. Flowchart Kolaborasi Algoritma Stemming dengan Metode Parsing Queries to_tsquery.

Berikut penjelasan dari alur diagram pada gambar 1:

3.1. Stemming nama-nama program studi pada basis data.

Semua nama program studi yang sudah ada didalam basis data, dilakukan stemming agar mendapatkan leksem dari setiap kata dari nama program studi tersebut.



Gambar 3. Diagram Proses Stemming Data Program Studi dan Simpan dalam Basis Data.

Tabel 2. Stemming Nama Program Studi di Basis Data

No.	Nama Program Studi	Jenjang	Stemming
1	Pengolahan dan Penyimpanan Hasil Perikanan	D4	olah dan simpan hasil ikan
2	Pengolahan Hasil Laut/Perikanan	D3	olah hasil laut ikan
3	Penanganan Krisis Kebencanaan	D3	tangan krisis bencana
4	Manajemen Pertahanan Matra Udara	D4	manajemen tahan matra udara
5	Administrasi Perkantoran Keimigrasian	D3	administrasi kantor imigrasi
6	Pengelolaan Hasil Perkebunan	D4	kelola hasil kebun
7	Studi Kependudukan dan Pencatatan Sipil	D4	studi duduk dan catat sipil
8	Pembangunan Ekonomi Kewilayahan	D4	bangun ekonomi wilayah

3.2. Stemming nama program studi yang diusulkan.

Dilakukan Stemming Nama Program Studi yang diusulkan oleh perguruan tinggi menggunakan Library Sastrawi PHP. Library ini sudah mengadopsi Algoritma Nazief & Andriani.

Tabel 3. Pseudocode Stemming Menggunakan Library Sastrawi

Pseudocode
BEGIN INPUT --> text-search GET --> new instance Sastrawi\Stemmer GET method --> createStemmer OUTPUT --> stem(text-search) END

```
$txtSearch = $this->input->post('txtSearch');

$stemmerFactory = new \Sastrawi\Stemmer\StemmerFactory();
$stemmer  = $stemmerFactory->createStemmer();
$result_stem = $stemmer->stem($txtSearch);
```

Gambar 4. Kode Stemming Menggunakan Library Sastrawi

3.3. Parsing hasil stemming untuk menambahkan operator & (AND).

Hasil stemming nama program studi yang diusulkan seterusnya dilakukan parsing untuk menyisipkan operator & (AND). Karena fungsi to_tsquery memerlukan operator AND untuk mencari pola kata pada field stemming pada basis data.

```php
function getParsingText($text)
{
  $tsquery = '';
  $data = explode(' ', $text);
  for($x=0;$x<sizeof($data);$x++)
  {
      if($x != sizeof($data)-1) {
          $tsquery .= $data[$x].' & ';
      } else {
        $tsquery .= $data[$x];
      }
  }
  return $tsquery;
}
```

Gambar 5. Parsing Text dan Menyisipkan Operator & (AND)

3.4. Gunakan hasil parsing pada fungsi to_tsquery.

Parsing text yang sudah disisipkan operator & (AND), digunakan pada fungsi to_tsquery

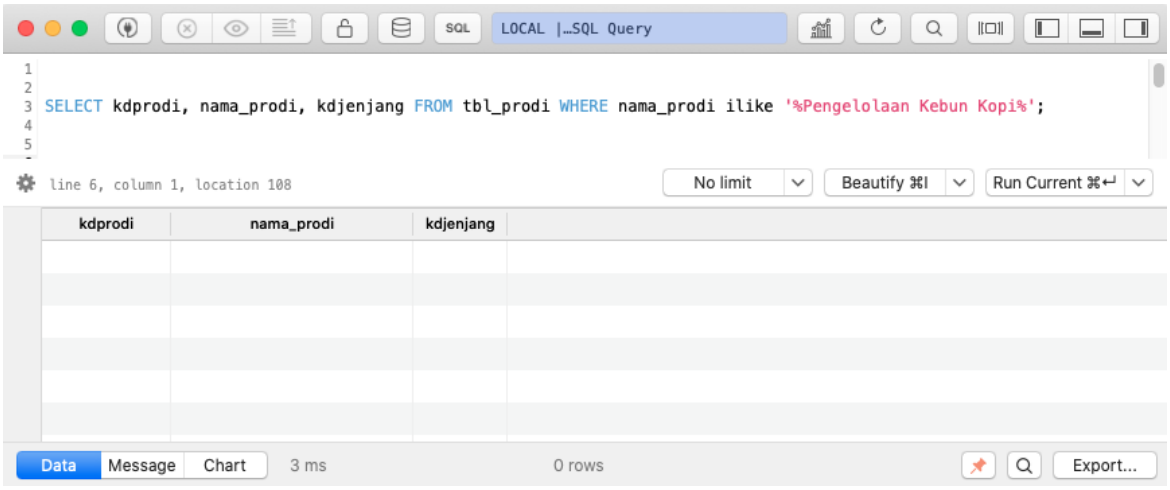
```
select *
    from tbl_prodi
        where stemming @@ to_tsquery('kelola & hasil & kebun');
```

4. HASIL DAN PEMBAHASAN

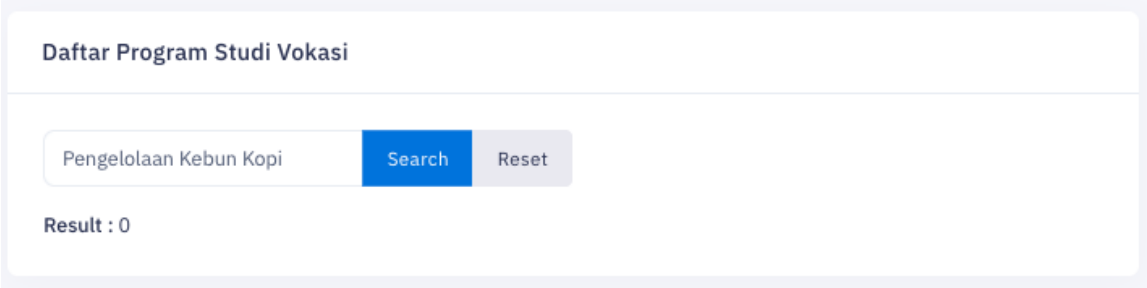
Dari hasil kolaborasi Parsing Query dengan Library Sastrawi (Algoritma Nazief & Andriani), didapatkan data yang memiliki kemiripan dalam basis data.

4.1. Menggunakan Query ILIKE sebelum menggunakan Parsing Query

Query ILIKE dengan operator percent % sering digunakan oleh para developer untuk filter atau mencari kata atau kalimat pada basis data, berikut ilustrasi penggunaan Query ILIKE.



Gambar 6. Data tidak ditemukan dengan menggunakan ILIKE operator % (percent).



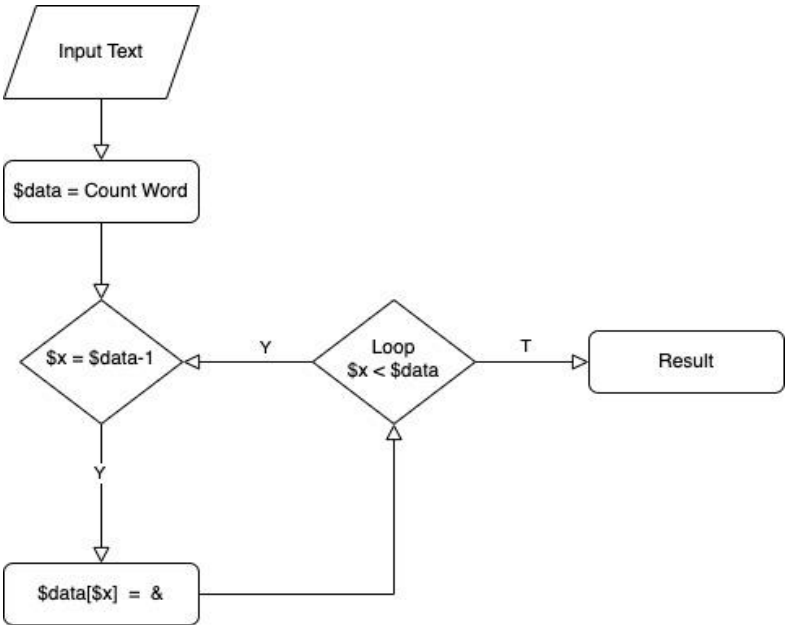
Gambar 7. Tampilan web hasil pengguna ILIKE dengan operator % (percent)

4.2. Menggunakan Parsing Query to_tsquery dengan operator & (AND)

Pada gambar 6 dan 7 dapat kita lihat tidak ditemukannya nama program studi yang dicari atau diusulkan, hasil ini perguruan tinggi menganggap program studi tersebut belum ada dan dapat diusulkan. Mengatasi permasalahan ini, penulis menggunakan parsing query kemudian diimplementasikan dengan fungsi to_tsquery, tetapi sebelumnya nama program studi yang diinputkan harus terlebih dahulu dinormalkan menjadi kata dasar (root word) menggunakan Librari Sastrawi (Algotima Nanief & Andirani).

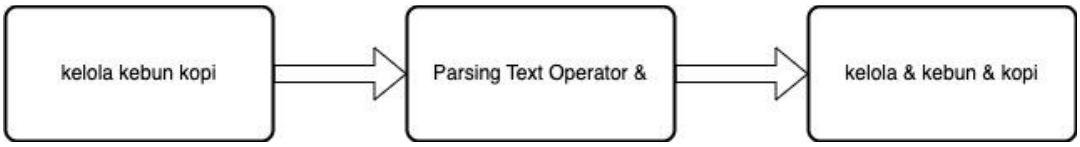
Gambar 8. Diagram Alir Stemming Kata Pengelolaan Kebun Kopi menjadi kelo kebun kopi

Setelah stemming nama program studi, dilakukan parsing text dari hasil stemming untuk menyisipkan operator & (AND) agar dapat digunakan pada parsing query to_tsquery.



Gambar 9. Diagram Alir Parsing Text

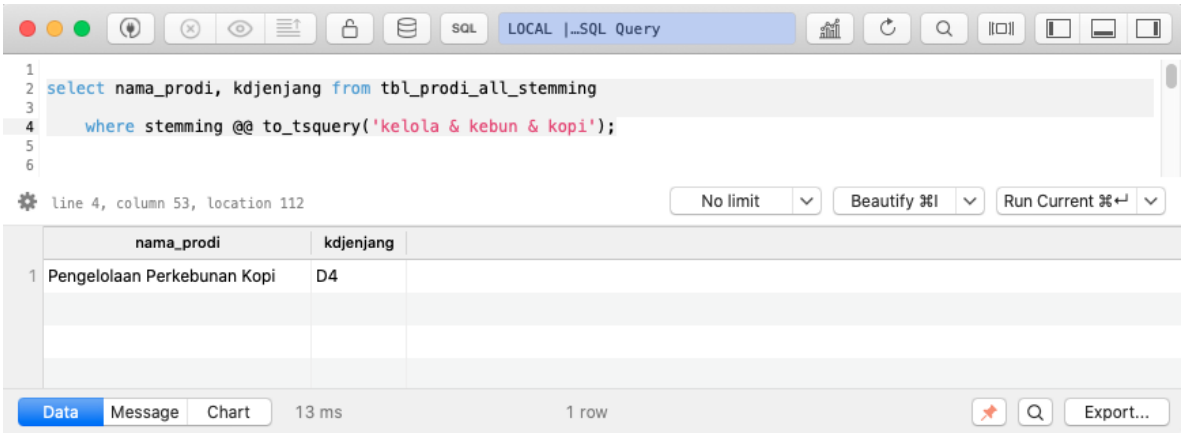
Berikut gambar dibawah ini flow parsing text untuk menambahkan operator & (AND)



Gambar 10. Parsing Text dengan Operator & (AND)

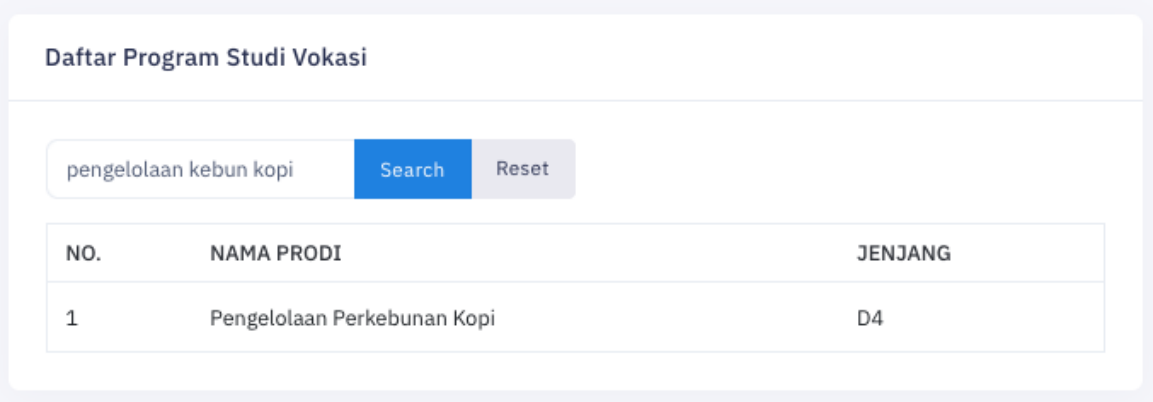
4.3. Implementasi

Berikut penggunaan secara keseluruhan pada aplikasi pengusulan program studi baru atau disebut juga dengan Nomenklatur Nama Program Studi.

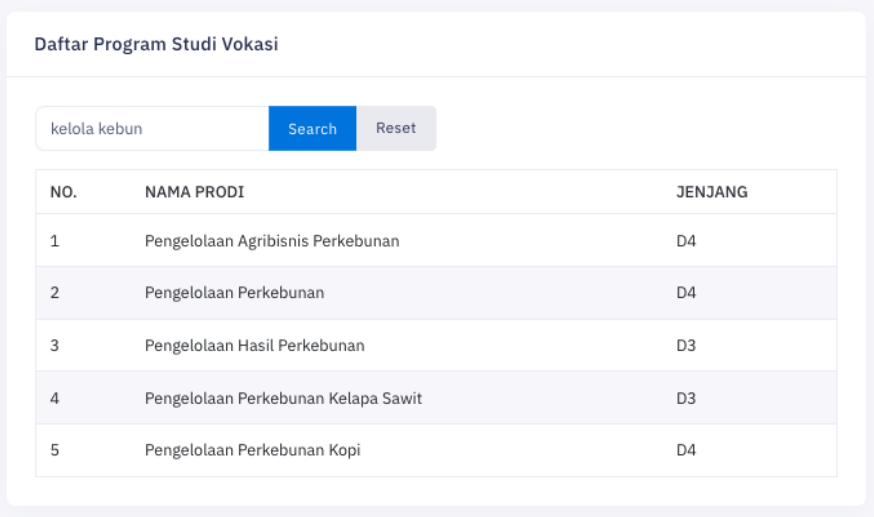


Gambar 11. Hasil Parsing Query menggunakan to_tsquery

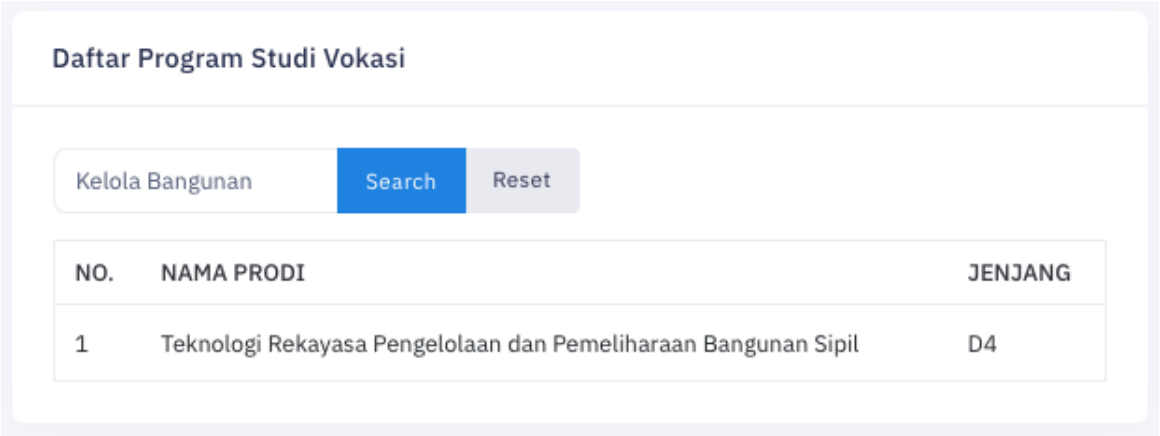
Pada gambar 11 menjelaskan bahwa terdapat kemiripan nama program studi yang sedang dicari. Dengan hasil ini dapat diimplementasi kan pada halaman web aplikasi.



Gambar 12. Hasil Parsing Queries Pada Halaman Web



Gambar 13. Parsing Queries Menggunakan Leksem atau Kata Dasar



Gambar 14. Contoh Lain Parsing Queries Menggunakan Leksem atau Kata Dasar

5. KESIMPULAN

Sebelum menggunakan metode Parsing Query, banyak data tidak ditemukan karena filter data menggunakan metode konvensional dalam kasus ini menggunakan operator ILIKE dengan pola *wildcard character* % (percent). Metode tersebut banyak digunakan pada aplikasi untuk searching data dan ditampilkan dalam bentuk tabel. Data dalam bentuk kata yang sudah mengalami perubahan dari akar kata (leksem) terkadang tidak terbaca oleh query menggunakan operator ILIKE.

Dengan mengkolaborasikan Algoritma Stemming dengan Parsing Queries dapat memberikan hasil nama program studi yang mirip. Kalimat atau nama program studi yang diinput terlebih dahulu dicari leksem katanya menggunakan algoritma stemming. kemudian dilakukan Parsing kalimat tersebut agar dapat dibaca oleh operator to_tsquery(). Ketika di implementasikan akan mendapatkan kalimat atau kata yang dicari, sehingga mengurangi kesalahan oleh sistem. Dengan ketepatan atau kemiripan nama program studi yang ada di dalam basis data, perguruan tinggi tidak perlu mengusulkan penambahan nama baru, dan tidak menghabiskan waktu dalam pembuatan proposal pengusulan nomenklatur program studi baru tersebut.

REFERENSI

[1] A. Gelbulkh, "Natural Language Processing," in *Fifth International Conference on Hybrid Intelligent Systems (HIS'05)*, Rio de Janeiro, Brazil, 2005, p. 6. doi: 10.1109/ICHIS.2005.79.

[2] F. Z. Tala, "A Study of Stemming Effects on Information Retrieval in Bahasa Indonesia," *M.Sc. Thesis, Append. D*, vol. pp, pp. 39–46, 2003.

[3] K. Divya, B. S. Siddhartha, N. M. Niveditha, and B. M. Divya, "An Interpretation of Lemmatization and Stemming in Natural Language Processing," *J. Univ. Shanghai Sci. Technol.,* vol. 22, no. 10, p. 351, 2020, [Online]. Available: https://www.researchgate.net/publication/348306833

[4] J. Asian, H. E. Williams, and S. M. M. Tahaghoghi, "Stemming Indonesian: A Confi x-Stripping Approach," *Conf. Res. Pract. Inf. Technol. Ser.*, vol. 38, no. September 2018, pp. 307–314, 2005, doi: 10.1145/1316457.1316459.

[5] D. Wahyudi, T. Susyanto, and D. Nugroho, "Implementasi Dan Analisis Algoritma Stemming Nazief & Adriani Dan Porter Pada Dokumen Berbahasa Indonesia," *J. Ilm. SINUS*, vol. 15, no. 2, pp. 49–56, 2017, doi: 10.30646/sinus.v15i2.305.

[6] S. Suhada and S. Bahri, "Implementasi Algoritma Rabin Karp Dan Stemming Najief Andriani Untuk Deteksi Plagiarisme Dokumen," *Swabumi*, vol. 5, no. 1, pp. 84–89, 2017, [Online]. Available: https://ejournal.bsi.ac.id/ejurnal/index.php/swabumi/article/view/1776

[7] A. C. Herlingga, I. P. E. Prismana, D. R. Prehanto, and D. A. Dermawan, "Algoritma Stemming Nazief & Adriani dengan Metode Cosine Similarity untuk Chatbot Telegram Terintegrasi dengan E-layanan," *J. Informatics Comput. Sci.*, vol. 2, no. 01, pp. 19–26, 2020, doi: 10.26740/jinacs.v2n01.p19-26.

[8] A. Jelita, "Effective Techniques for Indonesian Text Retrieval," *Ph.D Thesis*, pp. 1–286, 2007, [Online]. Available: https://researchbank.rmit.edu.au/view/rmit:6312

[9] A. Z. Arifin, P. Adhi, K. Mahendra, and H. T. Ciptaningtyas, "Enhanced Confix-Stripping Stemmer and Ants Algorithm for Classifying News Document in Indonesian Language," *5th Int. Conf. Inf. Commun. Technol. Syst.*, no. April 2014, pp. 149–158, 2009.

[10] A. D. Tahitoe and D. Purwitasari, "Enhanced Confix Stripping Stemmer," pp. 1–15, 2010.

[11] PostgreSQL, "PostgreSQL Documentation 15, Chapter 12, 'Full Text Search,'" 2022. https://www.postgresql.org/docs/current/textsearch-intro.html