IMPLEMENTASI *ABSTRACT SYNTAX TREE* PADA PENILAIAN KODE PROGRAM BAHASA PEMROGRAMAN C++

[1]Muhamad Mustamiin, [2]Iryanto, [3]Eka Ismantohadi
[1,2,3]Teknik Informatika, Politeknik Negeri Indramayu
E-mail: [1]mustamiin@polindra.ac.id

ABSTRACT

Online Judge System (OJS) has been widely used in programming learning processes, especially at the university level. The assessment process is carried out by comparing the results of source code execution based on pre-arranged test cases. In OJS there are limitations, the assessment is only based on right and wrong, it cannot make a comprehensive assessment based on a certain grade. There needs to be another alternative in determining the assessment of the results of OJS, such as using the process of checking the contents of the textual source code itself, not just the results of its execution. Assessment of the source code by measuring the similarity between the text and structure of the student source code and the answer key source code can be another alternative in measuring the accuracy of the source code assessment. In this article, the authors measure the text source code using cosine similarity and the structural source code using the jaccard method. From the test results it was found that the completion of the text between the answer source code and the answer key could not fully represent the appearance because many parts of the source code, especially the C++ programming language, needed to be studied in depth based on its structure. The application of the program code structure is carried out by creating an Abstract Syntax Tree (AST) which is very representative of the source code.

Keywords : online judge system, abstract syntax tree, jaccard, assessment, source code

ABSTRAK

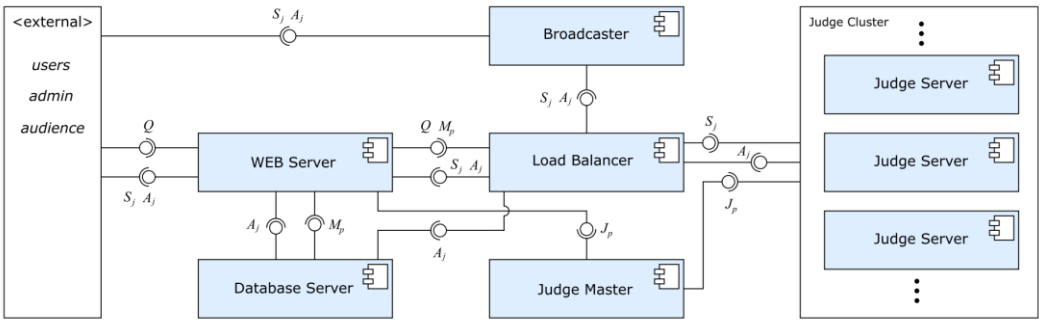
Online Judge System (OJS) telah banyak digunakan dalam proses pembelajaran pemrograman khususnya pada tingkat perguruan tinggi. Proses asesmen dilakukan dengan melakukan komparasi hasil dari eksekusi kode program berdasarkan test case yang telah diatur sebelumnya. Dalam OJS terdapat keterbatasan yaitu penilaian hanya berpatokan pada benar dan salah saja, tidak dapat menilai secara komprehensif berdasarkan grade tertentu. Perlu adanya alternatif lain dalam menentukan ketepatan penilaian dari hasil OJS, seperti menggunakan proses pengecekan dari isi tekstual kode program itu sendiri bukan hanya dari hasil eksekusinya. Penilaian kode program dengan mengukur kemiripan antara teks maupun struktur kode program mahasiswa dan kode program kunci jawaban dapat menjadi alternatif lain dalam mengukur ketepatan penilaian kode program. Pada artikel ini penulis melakukan pengukuran kemiripan teks kode program menggunakan cosine similarity dan kemiripan struktur kode program dengan menggunakan metode jaccard. Dari hasil ujicoba didapatkan bahwa perhitungan kemiripan teks antara kode program jawaban dan kunci jawaban belum dapat merepresentasikan kemiripan seutuhnya karena banyak bagian dari kode program khususnya bahasa pemrograman c++ yang perlu ditelaah secara mendalam berdasarkan strukturnya. Penerapan struktur kode program yang dilakukan dengan membuat Abstract Syntax Tree (AST) sangat representatif terhadap penilaian kode program.

Kata Kunci: *online judge sistem, abstrak sintaks tree, jaccard,* penilaian, kode program

PENDAHULUAN

Online judge system (OJS) adalah perangkat lunak yang dapat mengkompilasi, mengeksekusi, dan mengevaluasi kode program yang dikirimkan oleh pengguna. OJS awalnya hanya digunakan dalam kompetisi pemrograman (Zhou et al., 2018) . Namun, dengan seiring berjalannya waktu OJS ini telah terbukti memiliki efisiensi yang tinggi dalam proses pengkoreksian kode program maka sekarang sudah banyak digunakan dalam proses pembelajaran pemrograman, khususnya pada tingkat perguruan tinggi. Pada

Gambar 1 ditunjukkan sebuah bagan dari komponen-komponen utama dari sebuah OJS, diantaranya: *judge server, judge master, load balancer, broadcaster, database server* dan *web server*.



Gambar 1. Komponen Utama OJS (Watanobe et al., 2022)

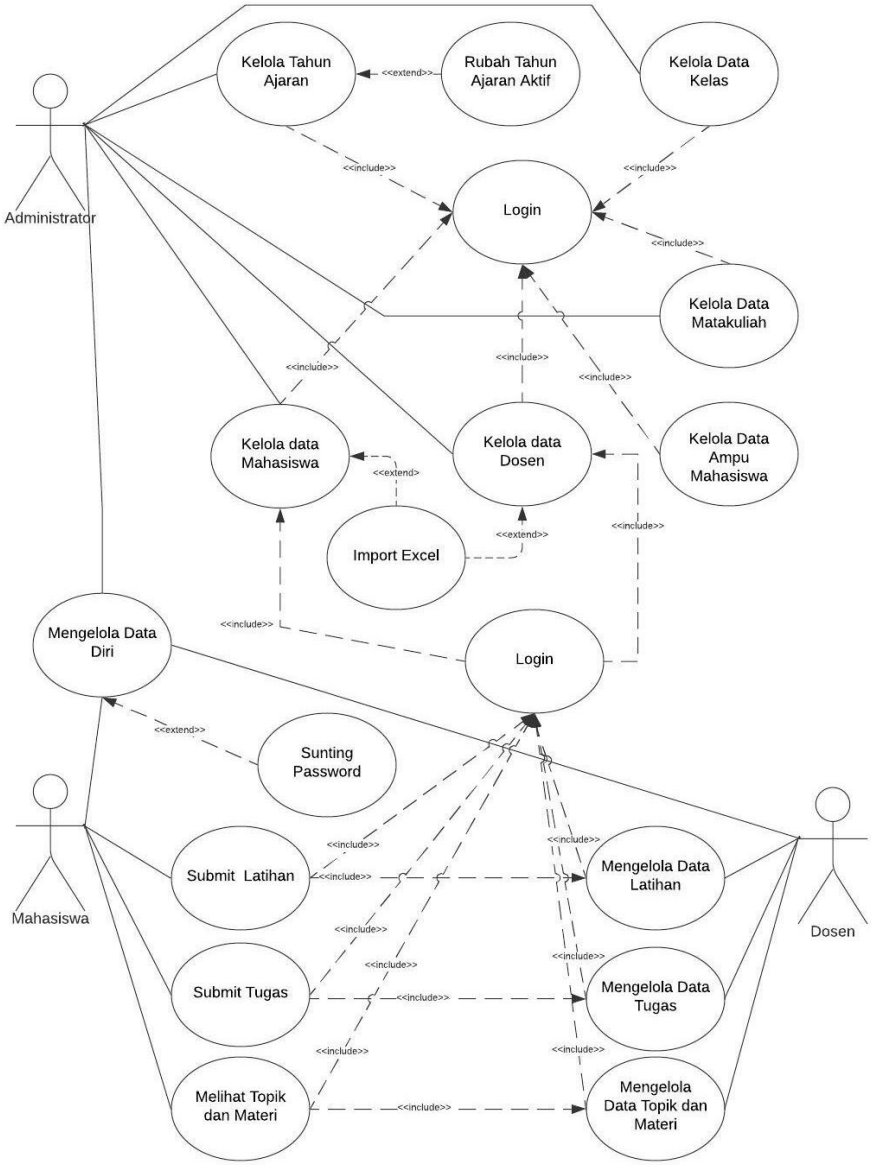
Selain komponen utama yang telah disebutkan pada Gambar 1, ada pula komponen pendukung yang menjadi bagian penting dalam proses evaluasi pembelajaran pemrograman yang jika dilihat pada Gambar 1 merupakan bagian "external". Salah satu contoh sistem pendukung tersebut adalah sistem evaluasi pemebelajaran pemrograman yang telah dikembangkan oleh Mustamiin (2021) yang terdiri dari beberapa fitur utama, diantaranya: Administrator dapat mengelola data mahasiswa, dosen, matakuliah, dan kelas. Dosen dapat mengelola data materi kuliah, latihan, dan tugas. Sementara mahasiswa dapat melihat materi, mengerjakan latihan dan mengumpulkan tugas seperti yang terlihat pada Gambar 2.

Penilaian kode program dalam OJS dilakukan dalam beberapa tahapan, yaitu submission, asesmen, dan skoring. Proses asesmen dilakukan dengan melakukan komparasi hasil dari eksekusi kode program berdasarkan *test case* yang telah diatur sebelumnya sesuai dengan soal atau permasalahan yang dikerjakan. Pada implementasinya, semakin banyak jumlah *test case* maka akan berpengaruh pada performa dari OJS karena akan semakin banyak pula proses eksekusi kode program yang dilakukan, tentunya ini akan semakin membutuhkan sumber daya server dan waktu dalam penyelesaian eksekusi kode program tersebut.

Di sisi lain penggunaan *test case* yang kurang memadai akan membuat hasil penilaian OJS menjadi kurang tepat karena ada kemungkinan terjadinya kondisi dimana kode program yang dikumpulkan mahasiswa belum benar 100% melainkan hanya kebetulan lolos dengan beberapa *test case* yang telah disediakan oleh dosen. Selain itu, untuk beberapa soal studi kasus yang baru, yang memang dibuat secara mandiri oleh dosen menyebabkan perlu adanya usaha untuk mengidentifikasi *test case* yang dapat mengakomodir segala kemungkinan yang dapat terjadi dari hasil eksekusi program, dimana proses identifikasi *test case* tersebut memerlukan waktu yang cukup banyak.

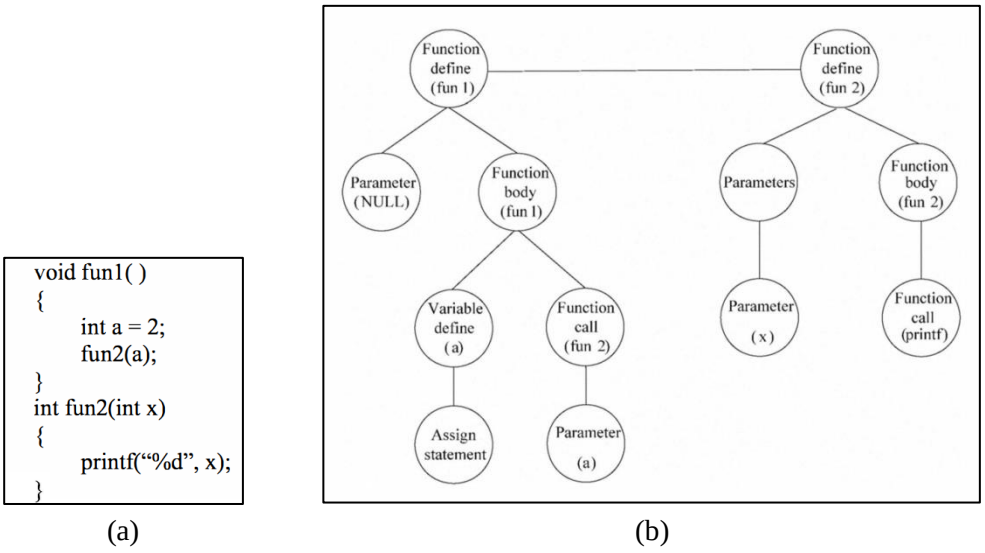
Kelemahan lain dari sistem OJS pada umumnya adalah OJS tidak dapat memeriksa apakah kode program mengandung beberapa poin sintaks pemrograman tertentu sesuai ketentuan soal atau studi kasus permasalahan terkait dengan progres pembelajaran yang telah atau sedang berlangsung (Zhou et al., 2018). Misalnya, jika ada soal atau studi kasus yang diberikan oleh Dosen meminta mahasiswa menyelesaikan permasalahan dengan menggunakan perulangan "for" bukan dengan "while" semata-mata untuk mengetahui tingkat pemahaman mahasiswa dalam materi perulangan "for". Namun, OJS pada

umumnya belum memiliki fitur untuk mendeteksi secara tekstual kode program tersebut. Oleh karena itu, kekurangan ini membuat mahasiswa hanya menggunakan sintaks yang menurut mereka "mudah" dalam menyelesaikan dan menyesuaikan *test case* yang ada, tanpa memperhatikan kriteri kode program yang harusnya digunakan pada soal tersebut.



Gambar 2. Use Case Sistem Evaluasi Pembelajaran Pemrograman (Mustamiin et al., 2021)

Dari beberapa penjelasan diatas, diketahui bahwa perlu adanya alternatif lain dalam menentukan ketepatan penilaian dari hasil OJS, seperti menggunakan proses pengecekan dari isi tekstual kode program itu sendiri bukan hanya dari hasil eksekusinya. Penilaian kode program dengan mengukur kemiripan antara teks maupun struktur kode program yang dikumpulkan oleh mahasiswa dan kode program kunci jawaban dari dosen dapat menjadi alternatif lain dalam mengukur ketepatan penilaian kode program.



Gambar 3. (a) contoh kode program, (b) contoh hasil kode program menjadi format *Abstract Syntax Tree* (Cui et al., 2010)

AST fokus pada faktor struktur sintaks dari kode program bukan pada level teks karena yang dibutuhkan adalah informasi struktur data dari sebuah kode program. Proses membuat AST dibagi menjadi empat (Cui et al., 2010), yaitu: 1) Pra-proses kode program. Misalnya, jika pernyataan "INTEGER" muncul di file kode program, semua pernyataan setelahnya diproses dengan mengganti "INTEGER" dengan "int". 2) Analisis leksikal dari kode program. Proses ini diselesaikan oleh Lex (kompiler Lexical), dan menghasilkan penganalisa leksikal yang membaca informasi kode program karakter demi karakter. 3) Analisis sintaks dari kode program. Proses ini diselesaikan oleh Y acc (Yet Another Compiler Compiler). Proses ini menghasilkan parser sintaks yang dapat menerima token yang dikembalikan dari Lex, lalu mencocokkan urutan token tertentu sesuai dengan aturan. 4) Pembuatan pohon sintaksis. Ketika urutan token tertentu cocok dengan aturan tertentu (seperti definisi fungsi), kemudian akan mengalokasikan ruang memori, menghasilkan simpul pohon sintaksis, dan menyimpan jenis simpul yang sesuai, serta posisi dalam kode program.

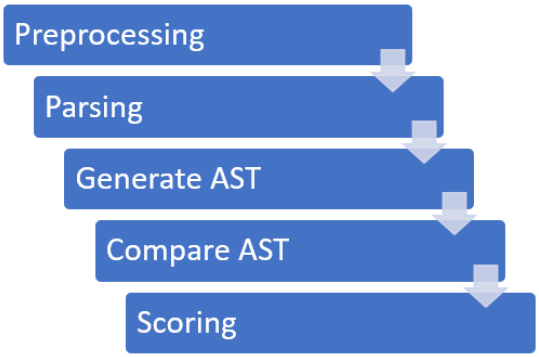
Pada artikel ini penulis melakukan pengukuran kemiripan teks kode program menggunakan *cosine similarity* dan kemiripan struktur kode program dengan menerapkan parsing kode program menjadi AST yang kemudian diukur kemiripannya dengan metode *jaccard similarity*.

METODE

Metode Penelitian yang dilakukan dalam penelitian ini terlihat pada Gambar 4, teridiri dari:

1) Preprocessing, tahapan awal untuk penyesuaian input dari teks kode program sebelum di parsing menjadi AST.
2) Parsing, pada tahap ini setiap aturan pada bahasa pemrograman c++ diterapkan dalam pengecekan setiap baris atau blok kode program.
3) Generate AST, hasil dari parsing kemudian dibuat sebuah file AST yang berformat *JavaScript Object Notation* (JSON).

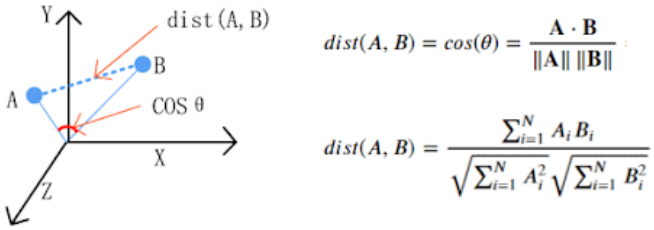
4) Compare AST, pada tahap ini AST dari kunci jawaban akan dibandingkan dengan AST dari jawaban kode program mahasiswa.
5) Scoring, ini adalah tahap akhir dimana dari hasil proses komparasi AST sebelumnya kemudian dihitung menggunakan metode Jaccard Similarity untuk menentukan kemiripan kode program yang akan menjadi representasi nilai jawaban kode program tersebut.



Gambar 4. Metode Penelitian

Dari berbagai metode dalam menentukan kemiripan teks penulis akan menggunakan metode *string based*, keuntungan dari metode berbasis string adalah mudah dihitung. *String based* yang mencakup pada urutan string dan komposisi karakter yang mengukur kemiripan atau perbedaan (jarak) antara dua string teks untuk perkiraan pencocokan atau perbandingan string. Perbedaan antara metode berbasis frasa dan metode berbasis karakter terletak pada unit dasarnya metode berbasis frase adalah kata frase, dan metode utama yang biasa diguanakan adalah koefisien dice, Jacard, dan sebagainya (Wang & Dong, 2020).

Perhitungan kemiripan berdasarkan string sederhana dan mudah diimplementasikan. Selain itu, ini juga dapat memberikan hasil yang baik untuk beberapa teks dengan kinerja yang baik melalui perbandingan tingkat karakter. Namun, kekurangannya adalah untuk dua teks yang makna kalimatnya sangat mirip, kemiripan yang dihitung berdasarkan string tidak dapat menangkap kemiripan semantik kedua teks maupun semantik leksikal kedua teks tersebut.



Gambar 5. Ilustrasi dan Rumus *Cosine Similarity* (datascientest.com, 2022)

Pada penelitian "Sistem koreksi jawaban uraian singkat otomatis menggunakan metode cosine similarity dan query ekspansion" dapat diketahui bahwa metode *cosine similarity* sangat cocok digunakan dalam menilai jawaban esai dengan akurasi mencapai 92% (Anam & others, 2020).

Jaccard similarity didefinisikan sebagai ukuran persimpangan dibagi dengan ukuran gabungan dari dua himpunan. Jaccard menyelesaikan kemiripan melalui himpunan; ketika teks relatif panjang, kemiripannya akan lebih kecil. Oleh karena itu, ketika Jaccard digunakan untuk menghitung kemiripan, biasanya dinormalisasi terlebih dahulu. Untuk bahasa Inggris, kata-kata dapat direduksi menjadi akar yang sama, dan untuk bahasa Cina, kata-kata dapat direduksi menjadi sinonim (Wang & Dong, 2020).

$$S(S_a, S_b) = \frac{S_a \cap S_b}{S_a \cup S_b}$$

HASIL DAN PEMBAHASAN

```
{
  VariableStatement: [
    [ 'int', 'a', null ],
    [ 'int', 'b', null ],
    [ 'int', 'hasil', null ]
  ]
},
{
  ExpressionStatement: { AssignmentExpression: [ '=', 'hasil', [ '+', 'a', 'b' ] ] }
},
{ OutputStatement: [ 'hasilnya adalah ', 'hasil' ] },
{ VariableStatement: [ [ 'int', 'hasil_pengurangan', null ] ] },
{
  ExpressionStatement: {
    AssignmentExpression: [ '=', 'hasil_pengurangan', [ '-', 'a', 'b' ] ]
  }
},
{ OutputStatement: [ 'hasilnya adalah ', 'hasil_pengurangan' ] }
```

Gambar 6. Hasil AST format JSON dari contoh kode program

Merujuk pada capaian pembelajaran mata kuliah algoritma dan pemrograman pada Jurusan Teknik Informatika Politeknik Negeri Indramayu serta merujuk pada buku algoritma dan pemrograman (Kristanto, 2003), penerapan *Abstract Syntax Tree* (AST) dibagi menjadi beberapa klasifikasi statemen pada bahasa pemrograman c++, diantaranya: Statement list in c++: *Variable statement, ConstantaStatement, If statement, Switch statement, Block statement, Void function statement, ReturnStatement, Iteration statement, ForStatement, WhileStatement, DoWhileStatement, Continue statement, Input statement, Output statement, Expression statement, AssignmentExpression*, dan *Assignment Operator*.

Pada Gambar 6 adalah contoh dari hasil deteksi struktur kode program c++ menjadi sebuah AST yang berformat JSON. Hasil parsing kode program jawaban mahasiswa menjadi AST kemudian dibandingkan dengan hasil parsing kode program kunci jawaban dari dosen.

Untuk penerapan perhitungan kemiripan dengan metode cosine similarity, penulis mengimplementasikannya dengan menggunakan fungsi dari *library* "string-cosine-similarity" pada npmjs.com (Jain, 2019). Sementara untuk implementasi perhitungan berdasarkan hasil parsing AST yaitu dengan perhitungan metode jaccard menggunakan fungsi dari *library* "set-distance" pada npmjs.com (Shevade, 2019).

Tabel 1. Hasil penilaian kode program 1

NO FILE	MANUAL	ONLINE JUDGE	COSINE	JACCARD
1	100	0	100	100
2	70	100	87,6	53,71
3	50	100	23,48	55,08
4	70	100	66,06	53,71
5	70	100	97,33	79,36
6	70	100	76,65	53,71

Tabel 2. Hasil penilaian kode program 2

NO FILE	MANUAL	ONLINE JUDGE	COSINE	JACCARD
1	80	100	99,04	86,67
2	70	100	98,62	76,25
3	100	100	100	100
4	100	100	100	100
5	90	0	97,62	80,56
6	100	100	99,01	97,22

Merujuk pada studi kasus soal mata kuliah algoritma dan pemrograman Jurusan Teknik Informatika Politeknik Negeri Indramayu yang terdapat pada sistem evaluasi pembelajaran (Mustamiin et al., 2021). Pada Tabel 1 dan Tabel 2 merupakan hasil perhitungan kemiripan kode program sebagai referensi penilaian pada evaluasi hasil pembelajaran. Berdasarkan analisis hasil perhitungan, didapatkan bahwa penggunaan online judge dalam penilaian dapat menjadi referensi penilaian, namun belum dapat dijadikan 100% landasan penilaian karena ada beberapa kemungkinan seperti kesalahan server dan kegagalan jaringan dapat mempengaruhi hasil penilaian. Sementara itu, formmat kode program dalam bentuk AST sebagai landasan dalam proses perhitungan kesamaan kode program memberikan hasil yang cukup mencerminkan kesesuaian kode program khususnya untuk studi kasus dengan klausul struktur kode program tertentu, misalnya: harus menyertakan variable dengan tipe data tertentu, adanya konstanta, ketentuan kondisional tertentu, dan lain sebagainya. Selain itu, ditemukan juga bahwa penggunaan nama identifier sangat berpengaruh terhadap kemiripan kode program berbasis tekstual dengan menggunakan metode *cosine similarity*.

KESIMPULAN

Implementasi *Abstract Syntax Tree* (AST) sebagai salah satu bagian proses penilaian kode program sangat tepat khususnya implementasi pada proses evaluasi pembelajaran pemrograman karena dapat menjadi pelengkap sistem online judge yang hanya mengacu pada parameter *test case* input dan output dari program. Dengan menerapkan AST, maka dosen dapat menganalisis hasil jawaban kode program mahasiswa secara mendetail dan dapat menemukan pola pemahaman mahasiswa dalam menyelesaikan suatu studi kasus pemrograman. Selain itu, mahasiswa juga dapat mengetahui bagian-bagian yang dapat menjadi umpan balik bagi mereka untuk menyelesaikan suatu permasalahan pemrograman khususnya pada mata kuliah algoritma dan pemrograman.

UCAPAN TERIMAKASIH

Tim penulis mengucapkan terima kasih kepada Pusat Penelitian dan Pengabdian kepada Masyarakat (P3M) Politeknik Negeri Indramayu yang telah mendukung penuh kegiatan penelitian ini sehingga terlaksana dengan baik.

DAFTAR PUSTAKA

Anam, M., & others. (2020). *Sistem koreksi jawaban uraian singkat otomatis menggunakan metode cosine similarity dan query ekspansion*. Universitas Islam Negeri Maulana Malik Ibrahim.

Cui, B., Li, J., Guo, T., Wang, J., & Ma, D. (2010). Code comparison system based on abstract syntax tree. *2010 3rd IEEE International Conference on Broadband Network and Multimedia Technology (IC-BNMT)*, 668–673.

datascientest.com. (2022, September 18). *Word2vec : NLP & Word Embedding*. Https://Datascientest.Com/Nlp-Word-Embedding-Word2vec.

Jain, K. (2019). *String Cosine Similarity*. Https://Www.Npmjs.Com/Package/String-Cosine-Similarity. https://www.npmjs.com/package/string-cosine-similarity

Kristanto, A. (2003). *Algoritma dan Pemrograman dengan C++*.

Mustamiin, M., Iryanto, I., Andi, M., & Ismantohadi, E. (2021). Pengembangan Sistem Manajemen Evaluasi Pembelajaran Terintegrasi Dengan Online Judge. *Ikraith-Informatika*, *5*(3), 64–71.

Shevade, V. (2019). *set-distance*. https://github.com/varad11/set-distance

Wang, J., & Dong, Y. (2020). Measurement of text similarity: a survey. *Information*, *11*(9), 421.

Watanobe, Y., Rahman, Md. M., Matsumoto, T., Rage, U. K., & Ravikumar, P. (2022). Online Judge System: Requirements, Architecture, and Experiences. *International Journal of Software Engineering and Knowledge Engineering*, *32*(06), 917–946. https://doi.org/10.1142/S0218194022500346

Zhou, W., Pan, Y., Zhou, Y., & Sun, G. (2018). The framework of a new online judge system for programming education. *Proceedings of ACM Turing Celebration Conference - China*, 9–14. https://doi.org/10.1145/3210713.3210721