CCGtown: Alat Anotasi Combinatory Categorial Grammar (CCG) Semi-otomatis Berbasis Web

Wisnu Adi Nurcahyo[1], Ade Romadhony[2]

[1,2]Fakultas Informatika, Universitas Telkom, Bandung
[1]nurcahyo@student.telkomuniversity.ac.id, [2]aderomadhony@telkomuniversity.ac.id

Alat anotasi yang tersedia untuk *combinatory categorial grammar* (CCG) masih sangat terbatas. Salah satu aplikasi dengan antarmuka grafis yang dapat digunakan adalah CCGweb. Untuk dapat menggunakan CCGweb, pengguna diharuskan untuk melakukan *setup* secara mandiri. Proses ini dapat menjadi keterbatasan bagi *annotator* karena *dependency* yang perlu dipersiapkan cukup banyak. Selain itu, CCGweb hanya dapat mengerjakan satu proyek anotasi CCG dalam satu waktu. Demikian itu, kami membangun alat anotasi CCG yang dapat langsung digunakan oleh pengguna tanpa perlu melakukan *setup* terlebih dahulu. Alat anotasi tersebut kami beri nama CCGtown. Dengan CCGtown, pengguna dapat membuat banyak proyek anotasi sekaligus dan telah dilengkapi fitur yang sebagian besarnya tersedia di CCGweb. CCGtown merupakan alat anotasi CCG alternatif yang dapat digunakan oleh *annotator* CCG.

Kata kunci: pemrosesan bahasa alami, combinatory categorial grammar, alat anotasi

Abstract

Annotation tools available for combinatory categorial grammar (CCG) are still limited. One of the annotation tool with a graphical interface that can be used is CCGweb. To be able to use CCGweb, users are required to setup independently. This process can be a limitation for annotators because of the many dependencies that need to be prepared. CCGweb also can only work on one CCG annotation project at a time. With that in mind, we built a CCG annotation tool that users can use right away without any setup required. We call the annotation tool CCGtown. With CCGtown, users can create multiple annotation projects at once and it has features that are mostly available on CCGweb. CCGtown is an alternative CCG annotation tool which can be used by CCG annotators.

Keywords: natural language processing, combinatory categorial grammar, annotation tool

1. Pendahuluan

Latar Belakang

CCGweb[1] merupakan alat anotasi *open source* berbasis web pertama yang dikembangkan khusus untuk memberikan anotasi CCG[3]. Fitur yang ditawarkan CCGweb cukup beragam mulai dari *dynamic annotation*, WYSIWYG (*what you see is what you get*), *lexical category constraint*, *span constraint*, *issue reporting* via layanan *web service* eksternal, hingga *adjudication support*. Selain itu, CCGweb dibangun untuk membangun *dataset* CCG *multilingual* yang artinya terdapat lebih dari satu bahasa untuk satu kalimat yang sama. Untuk dapat menggunakan CCGweb, pengguna harus melakukan instalasi manual secara mandiri dan dapat dilakukan baik di komputer pribadinya ataupun di layanan *hosting* maupun *cloud*. Adapun proyek anotasi yang dapat dikerjakan dalam satu waktu adalah satu buah proyek yang mana dapat memiliki satu bahasa atau lebih. Demikian itu, pengguna tidak dapat mengerjakan lebih dari satu proyek dan harus menyelesaikan proyek sebelumnya agar dapat memulai proyek baru. Alternatif lainnya adalah melakukan instalasi kembali sehingga pengguna akan memiliki lebih dari satu aplikasi CCGweb.

Untuk dapat menggunakan CCGweb, pengguna diharapkan telah mempersiapkan beberapa *dependency* yang diperlukan yaitu EasyCCG[2] (termasuk berkas modelnya), Elephant *tokenizer*[3] (termasuk

[1]https://github.com/texttheater/ccgweb
[2]https://github.com/ParallelMeaningBank/easyccg
[3]https://github.com/ParallelMeaningBank/elephant

berkas modelnya), berkas model UDPipe[4], Produce *build system*[5], dan Viasock[6]. Selain *dependency* tersebut, beberapa *dependency* lainnya relatif mudah untuk dipersiapkan sehingga dapat kita lewati. Kelima *dependency* yang diperlukan tersebut harus dipersiapkan secara mandiri. Hal ini menyulitkan bagi pengguna yang ingin menggunakan CCGweb di komputer pribadinya karena proses ini rawan mengalami *error* seperti perbedaan versi yang digunakan antar *dependency*-nya dan sebagainya. Kendala yang dialami oleh calon pengguna ketika melakukan *setup* juga dapat mengurangi minat calon kontributor untuk memberikan kontribusi bagi pengembangan CCGweb. Demikian itu, langkah *setup* yang perlu dilakukan sebaiknya dikurangi hingga sesedikit mungkin. Salah satu solusinya adalah dengan menambahkan perintah baru untuk melakukan otomasi proses *setup* seperti menyiapkan semua *dependency* yang diperlukan secara otomatis, melakukan instalasi DBMS[7] secara otomatis, membuatkan tabel-tabel yang diperlukan secara otomatis, dan seterusnya. Akibatnya, calon pengguna dan calon kontributor dapat langsung fokus pada tujuannya dalam menggunakan alat anotasi tersebut tanpa perlu direpotkan untuk melakukan *setup* yang panjang.

CCGtown[8] merupakan alat anotasi CCG alternatif yang dapat digunakan oleh *annotator*. CCGtown memiliki kemampuan untuk membuat banyak proyek anotasi sekaligus. Selain itu, CCGtown juga dapat langsung digunakan secara daring tanpa perlu melakukan instalasi terlebih dahulu. Untuk memudahkan calon kontributor, CCGtown menggunakan *web framework* populer yang dapat dipelajari oleh siapapun serta menyediakan perintah-perintah yang dapat digunakan untuk melakukan otomasi seperti contohnya pada proses *setup*-nya. CCGtown juga dipublikasikan sebagai *open source software* sehingga siapapun dapat melihat sumber kodenya, menambahkan fitur, mengurangi fitur, mengunggah CCGtown di *server* pribadinya, dan masih banyak lagi. Adapun fitur yang dimiliki CCGtown saat ini mirip dengan CCGweb hanya saja dengan beberapa penyesuaian. Pengguna dapat menambahkan banyak proyek sekaligus, dapat menambahkan kalimat yang ingin diberikan anotasi, dapat memberikan anotasi CCG, dapat menyunting CCG *derivation* secara langsung dengan konsep WYSIWYG, dapat melakukan *generate* CCG *derivation*, dan dapat melakukan *auto-assign* CCG *lexicon*. Fitur-fitur lainnya dapat ditambahkan di lain waktu.

Topik dan Batasannya

Anotasi berdasarkan KBBI merupakan sebuah *catatan* yang dibuat oleh pengarang atau orang lain untuk menerangkan, mengomentari, atau mengkritik teks karya sastra atau bahan tertulis lain. Dalam konteks pemrosesan bahasa alami, anotasi merupakan sebuah *catatan* yang digunakan untuk merepresentasikan suatu makna tertentu. Representasi tersebut umumnya sesuatu yang dapat "dipahami" oleh komputer. Sebagai contoh, pada kalimat "Pamungkas kemarin makan rendang" kita dapat memberikan anotasi "Pamungkas[ORANG] kemarin makan rendang[MAKANAN]". Maksud dari anotasi tersebut yaitu "Pamungkas" dalam kalimat tersebut merupakan representasi dari orang sebagai subjeknya dan "rendang" merupakan representasi dari makanan sebagai objeknya. Memberikan anotasi secara manual merupakan kegiatan yang melelahkan. Demikian itu, alat anotasi dikembangkan untuk membantu meringankan proses pemberian anotasi.

Alat anotasi untuk pemrosesan bahasa alami yang tersedia sejatinya sudah cukup banyak. Jenis, kemampuan, dan biaya masing-masing alat anotasi tersebut juga beragam. Sebagai contoh, tagtog[9] merupakan alat anotasi berbasis web yang dapat digunakan secara gratis maupun berbayar. Selain tagtog, prodigy[10] juga merupakan alat anotasi berbasis web tetapi tidak dapat digunakan secara gratis. Selain itu, prodigy mendukung lebih banyak tipe anotasi seperti Named Entity, POS Tagging, Dependency Parsing, dan lain-lain. Kendati banyaknya alat anotasi yang sudah tersedia, dukungan anotasi untuk Combinatory Categorial Grammar (CCG) belum banyak. Salah satu alat anotasi CCG dengan antarmuka grafis yang tersedia adalah CCGweb[11].

Anotasi CCG sebenarnya memiliki bentuk yang rumit. Anotasi CCG memiliki bentuk sintaktik dan bentuk semantik. Bentuk (S/N) yang akan dilihat pada bagian selanjutnya merupakan bentuk sintaktik dari CCG[4]. Adapun : $\lambda x.\lambda y.\ suka(y, x)$ yang akan dilihat pada bagian selanjutnya merupakan bentuk semantik dari CCG. Bentuk sintaktik CCG sejatinya juga dapat lebih kompleks ketimbang hanya memiliki bentuk (S/N) saja. Akan tetapi, CCGtown saat ini ekspektasinya hanya dapat digunakan untuk anotasi CCG yang bentuknya sederhana. Hal tersebut dikarenakan terbatasnya sumber *dataset* yang dapat

[4] https://ufal.mff.cuni.cz/udpipe
[5] https://github.com/texttheater/produce
[6] https://github.com/texttheater/viasock
[7] *database management system*
[8] https://github.com/wisn/ccgtown
[9] https://tagtog.net/
[10] https://prodi.gy/
[11] https://ccgweb.phil.hhu.de/

dijadikan sampel. Salah satu *dataset* yang dapat digunakan adalah CCGbank. Namun, CCGbank bukanlah *dataset* yang dapat dengan bebas diperoleh. Demikian itu, CCGtown menggunakan sampel yang tersedia secara terbuka saja seperti contoh kasus dari referensi yang digunakan, contoh anotasi CCG di halaman NLTK, dan sebagainya.

Fokus CCGtown pada Tugas Akhir ini adalah untuk memberikan alternatif alat anotasi CCG yang telah tersedia yaitu CCGweb. CCGtown menyuguhkan *development cycle* yang lebih baik dari CCGweb dan menyediakan perintah-perintah untuk melakukan berbagai macam prosesnya secara otomatis. Salah satunya adalah perintah untuk melakukan *setup*-nya. Karena keterbatasan waktu, beberapa fitur yang telah tersedia di CCGweb akan dihilangkan atau diganti. Tugas Akhir ini berupaya untuk menunjukkan bahwa alat anotasi CCG dapat dikembangkan dengan menggunakan *web framework* yang telah tersedia serta dapat juga menggunakan *deployment tool* yang telah tersedia. Berbanding terbalik dengan CCGweb yang tidak menggunakan *web framework* apapun serta menggunakan *build tools* kustom yang dibuat sendiri.

Tujuan

CCGtown diharapkan dapat menjadi alat anotasi CCG alternatif yang dapat digunakan secara langsung oleh pengguna tanpa perlu melakukan instalasi terlebih dahulu. CCGtown juga diharapkan dapat memberikan proses *development* dan proses *deployment* yang lebih baik agar calon kontributor dapat dengan mudah memberikan serta melakukan pengujian terhadap kontribusinya.

Organisasi Tulisan

Studi Terkait menjelaskan dasar-dasar materi yang perlu diketahui sebelum beranjak ke bagian selanjutnya. Bagian tersebut membahas apa itu Categorial Grammar (CG), apa itu Combinatory Categorial Grammar (CCG), penjelasan singkat mengenai Lambda Calculus yang digunakan oleh CCG sebagai semantiknya, kemudian penjelasan singkat mengenai CCGweb. **Sistem yang Dibangun** menjelaskan mengenai teknologi apa saja yang digunakan, mengapa menggunakan teknologi tersebut, seperti apa desain database dan sistemnya, serta menjelaskan mengenai *deployment* CCGtown. **Evaluasi** memberikan hasil pengujian dengan menggunakan black-box testing (state transition test).

2. Studi Terkait

Categorial Grammar

Categorial Grammar (CG) merupakan sebuah istilah yang mencakup beberapa formalisme terkait yang diajukan untuk sintaks dan semantik dari bahasa alami serta untuk bahasa logis dan matematis [9]. Karakteristik yang paling terlihat dari CG adalah bentuk ekstrim dari leksikalismenya di mana beban utama (atau bahkan seluruh beban) sintaksisnya ditanggung oleh leksikon. Konstituen tata bahasa dalam *categorial grammar* dan khususnya semua leksikal diasosiasikan dengan suatu *type* atau "*category*" (dalam *category theory*) yang mendefinisikan potensi mereka untuk dikombinasikan dengan konstituen lain untuk menghasilkan konstituen majemuk. *Category* tersebut adalah salah satu dari sejumlah kecil *category* dasar (seperti NP) atau *functor* (dalam *category theory*). Dalam hal ini, *category* dapat diartikan sebagai *syntactic type* dari suatu kata.

Secara formal, *syntactic type* didefinisikan sebagai himpunan bagian dari suatu *semigroup* M yang tunduk pada tiga operasi yaitu 1, 2, dan 3 dimana A, B, dan C merupakan himpunan bagian dari M [4]. Adapun $A \cdot B$ dibaca A *times* B, C/B dibaca C *over* B, dan $A\backslash C$ dibaca A *under* C. Selanjutnya, dapat dilihat bahwasannya untuk semua $A, B, C \subseteq M$ sehingga kita dapatkan 4 dan 5. Terakhir, persamaan 6 dapat diabaikan apabila dihadapkan dengan *multiplicative system* yang tidak asosiatif. Sementara itu, apabila *semigroup*-nya merupakan sebuah *monoid* dengan identitas 1 maka kita dapatkan 7 dimana $I = \{1\}$.

$$A \cdot B = \{x \cdot y \in M \mid x \in A \wedge y \in B\} \tag{1}$$

$$C/B = \{x \in M \mid \forall_{y \in B} x \cdot y \in C\} \tag{2}$$

$$A\backslash C = \{y \in M \mid \forall_{x \in A} x \cdot y \in C\} \tag{3}$$

$$
\begin{aligned}
\text{Pamungkas} &\vdash \text{NP} : \ pamungkas' \\
\text{Setyo} &\vdash \text{NP} : \ setyo' \\
\text{dan} &\vdash \text{CONJ} : \ \lambda x.\lambda y.\lambda f.\ (f\ x) \wedge (f\ y) \\
\text{menyukai} &\vdash (\text{S}\backslash\text{NP})/\text{NP} : \ \lambda x.\lambda y.\ suka(y, x) \\
\text{rendang} &\vdash \text{NP} : \ rendang'
\end{aligned}
$$

Gambar 1. Kamus yang memetakan token kata ke bentuk CCG *lexicon*-nya.

$$A \cdot B \subseteq C \quad \text{jika dan hanya jika} \quad A \subseteq C/B \tag{4}$$

$$A \cdot B \subseteq C \quad \text{jika dan hanya jika} \quad B \subseteq A\backslash C \tag{5}$$

$$(A \cdot B) \cdot C = A \cdot (B \cdot C) \tag{6}$$

$$I \cdot A = A = A \cdot I \tag{7}$$

Ada beberapa notasi berbeda untuk *category* dalam merepresentasikan *directional*-nya. Notasi yang paling umum digunakan adalah "*slash notation*" yang dipelopori oleh Bar-Hilel, Lambek, dan kemudian dimodifikasi dalam kelompok teori yang dibedakan sebagai tata bahasa "*combinatory*" *categorial grammar* (CCG). Sebagai contoh, *category* (S\NP)/NP merupakan suatu *functor* yang memiliki dua buah notasi *slash* yaitu \ dan /. Masing-masing notasi *slash* tersebut merepresentasikan *directionality* yang berbeda. Notasi *forward slash*, /, mengindikasikan bahwa argumen dari suatu *functor* X/Y ada di bagian kanan atau dengan kata lain Y. Adapun *backward slash*, \, mengindikasikan bahwa argumen dari suatu *functor* X\Y ada di bagian kiri atau dengan kata lain X. Demikian itu, penggunaan notasi *slash* yang tepat sangat penting dikarenakan hal ini dapat mempengaruhi konstituen dari hasil "kombinasi" *category*-nya.

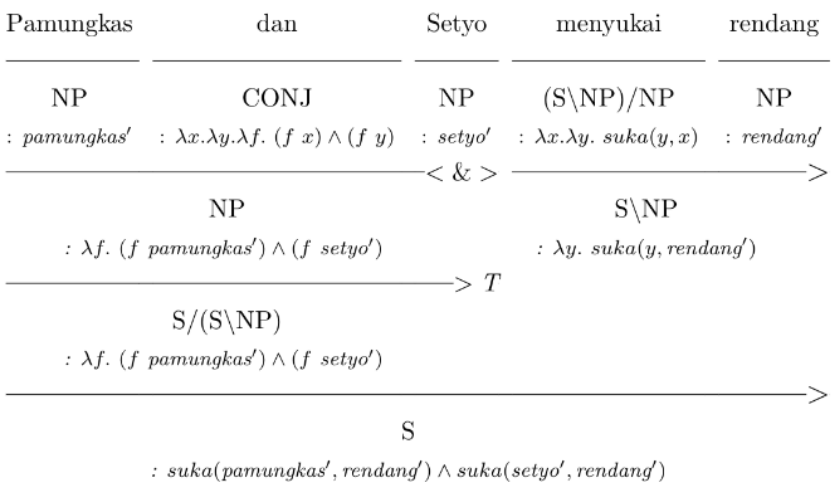
Combinatory Categorial Grammar

Combinatory Categorial Grammar (CCG) merupakan salah satu formalisme tata bahasa yang gaya aturannya diturunkan dari *categorial grammar* dengan beberapa penambahan aturan dan istilah baru [10]. Di CCG, *category* dapat dipasangkan dengan *semantic representation*. Dalam hal ini, *semantic representation* yang dimaksud adalah abstraksi fungsi lambda (dalam *lambda calculus*, *lambda function*). Sebagai contoh, *category* (S\NP)/NP dapat dipasangkan dengan fungsi lambda $\lambda x.fx$ sehingga dapat ditulis menjadi (S\NP)/NP : $\lambda x.fx$. Adapun pemetaan dari suatu token kata ke *category*-nya menggunakan notasi $\vdash$. Sebagai contoh, anggap saja kita memiliki kamus pemetaan seperti pada Gambar 1. Apabila kita memiliki kalimat "Pamungkas dan Setyo menyukai rendang", maka kita dapatkan:

Pamungkas	dan	Setyo	menyukai	rendang
NP	CONJ	NP	(S\NP)/NP	NP
$: pamungkas'$	$: \lambda x.\lambda y.\lambda f.\ (f\ x) \wedge (f\ y)$	$: setyo'$	$: \lambda x.\lambda y.\ suka(y, x)$	$: rendang'$

Ada beberapa operasi yang dapat dilakukan dalam CCG. *Operand* dari operasi yang dimaksud adalah *category*. Berdasarkan contoh di atas, akan ada tiga operasi yang dijalankan yaitu *coordination*, *forward application*, dan *type rising*. Untuk mendapatkan hasil yang diinginkan, kita lakukan *type rising* sebelum *forward application* di akhir. Sehingga, kita dapatkan Gambar 2. Berdasarkan hasil evaluasi tersebut, kita dapatkan *query* 8 yang diperoleh dari kalimat "Pamungkas dan Setyo menyukai rendang". Demikian itu, komputer dapat melakukan komputasi berdasarkan *query* yang telah diperoleh. Kegiatan tersebut merupakan apa yang disebut dengan CCG *parsing*. Untuk dapat melakukan parsing, CCG *lexicon* diperlukan. Untuk mendapatkan CCG *lexicon* kita dapat menggunakan CCG *supertagger* yang akan melakukan pelabelan suatu token kata ke CCG *lexicon* berdasarkan pemetaannya.

$$suka(pamungkas', rendang') \wedge suka(setyo', rendang') \tag{8}$$



Gambar 2. Contoh CCG *derivation* dengan operasi *coordination*, *forward application*, dan *type rising*.

Lambda Calculus

Lambda calculus (λ-*calculus*) merupakan sebuah formalisme yang dikembangkan oleh Alonzo Church sebagai alat yang digunakan untuk memahami konsep komputasi yang efektif [7]. Formalisme λ-*calculus* cukup populer dan bahkan dijadikan sebagai pondasi teori bagi paradigma pemrograman *functional programming*. Konsep utama dari λ-*calculus* adalah apa yang disebut dengan *expression*. Suatu *expression* dalam λ-*calculus* terdiri dari tiga bagian yaitu *lambda notation* (λ), *argument* (seperti a, b, c, x, dan lain-lain), dan *body* yang dipisahkan dengan tanda titik. Sebagai contoh, fungsi lambda $\lambda x.x$ merupakan sebuah fungsi identitas yang mengambil argumen x kemudian mengembalikan nilai x itu sendiri. Dalam hal ini, terlihat bahwa notasi λ merupakan sebuah penanda bagi suatu fungsi lambda. Kemudian, pengubah x setelah notasi λ merupakan argumen dari fungsi tersebut. Selanjutnya, tanda titik merupakan pemisah antara *head* dan *body* fungsi lambda. Terakhir, setelah tanda titik adalah *body* dari suatu fungsi lambda yang mana berupa *expression*.

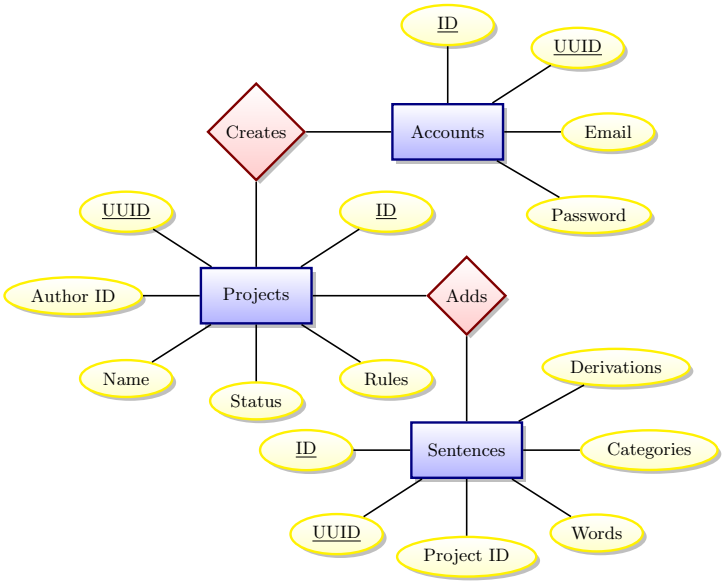
Untuk mempermudah pemahaman, λ-*calculus* dapat diperlakukan seperti fungsi tanpa nama. Sebagai contoh, fungsi lambda $(\lambda x.x + 5)$ apabila diberikan nilai 2 sehingga menjadi $(\lambda x.x + 5)2$ akan dievaluasi menjadi $\lambda(2).(2) + 5$. Demikian itu, nilai yang dikembalikan oleh fungsi tersebut adalah 7. Sama seperti fungsi pada umumnya, konsep ini bernama *substition* (substitusi). Memahami λ-*calculus* dirasa perlu berhubung dalam tugas akhir ini λ-*calculus* digunakan sebagai bentuk formal di *category* dalam konteks CCG *lexicon*. Meskipun λ-*calculus* tidak sesederhana yang dijelaskan sebelumnya, setidaknya memahami λ-*calculus* seperti ini sudah cukup untuk dapat membangun *supertagger* yang ada di tugas akhir ini.

CCGweb

CCGweb[12] merupakan *open source graphical annotation tool* pertama untuk CCG [3]. Aplikasinya berbasis web dan dibangun dengan menggunakan bahasa pemrograman Python, PHP, dan JavaScript. Fitur yang paling menarik dari *graphical annotation tool* adalah What You See Is What You Get (WYSIWYG) yang mana berupa kemampuan untuk me-*render* CCG *derivation* sesuai dengan apa yang kita lihat. Maksudnya, CCG *derivation* akan ditampilkan horizontal sesuai dengan panjang kalimatnya kemudian hasil *derivation*-nya ditampilkan vertikal seperti contoh pada Bagian 2..

Untuk dapat menggunakan CCGweb, kita perlu melakukan instalasi terlebih dahulu. Selanjutnya barulah kita dapat menambahkan kalimat-kalimat yang ingin dianotasi. Satu instalasi CCGweb hanya dapat digunakan untuk satu proyek anotasi sehingga apabila kita memiliki lebih dari satu proyek maka kita perlu melakukan instalasi CCGweb yang baru. Demikian itu, CCGtown[13] hadir dengan fitur *multi-project* dan tanpa perlu melakukan instalasi di komputer lokal karena aplikasinya *hosted* sehingga dapat diakses kapan pun.

[12] https://github.com/texttheater/ccgweb
[13] https://github.com/wisn/ccgtown



Gambar 3. Conceptual Entity Relationship Diagram (ERD) CCGtown

3. Sistem yang Dibangun

CCGtown dibangun dengan menggunakan bahasa pemrograman Python dan JavaScript. Adapun *framework* yang digunakan adalah Django. Versi awal CCGtown merupakan sebuah *proof-of-concept* dari *open source graphical annotation tool* berbasis web yang dilengkapi dengan fitur penganotasian semi-otomatis. Bahasa pemrograman Python digunakan karena sebagian besar *library* untuk CCG sudah tersedia di PyPi [14]. Salah satu *library* penting yang digunakan sebagai dasar dari fitur penganotasian semi-otomatis adalah NTLK [15]. Selanjutnya, Django digunakan untuk mempercepat proses pengembangan aplikasi. Adapun JavaScript digunakan untuk menjadikan CCGtown aplikasi berbasis web yang interaktif.

Alur kerja CCGtown pada umumnya adalah (1) pengguna melakukan registrasi, (2) pengguna melakukan *login* ke sistem, (3) pengguna membuat proyek baru, (4) pengguna menambahkan kalimat yang ingin dianotasi, (5) pengguna melakukan anotasi kemudian melakukan *generate* CCG *derivation* dan/atau melakukan modifikasi *derivation*-nya apabila diperlukan, dan (6) pengguna melakukan *export* setelah selesai melakukan anotasi. Alur kerja tersebut mempengaruhi desain sistem dari CCGtown. Salah satunya adalah desain dari *database* yang akan digunakan.

Desain Database

CCGtown menggunakan PostgreSQL sebagai DBMS[16]-nya. Hal ini karena PostgreSQL memiliki kemampuan untuk menyimpan struktur data JSON[17] sehingga memudahkan CCGtown untuk menyimpan format JSON dari CCG *derivation* yang telah dimanipulasi oleh pengguna melalui fitur *editable CCG derivation*. PostgreSQL juga memiliki banyak fitur lain termasuk di antaranya dukungan dari *non-relational database model* (seperti *multi-model graph*) sehingga apabila di waktu yang akan datang CCGtown memerlukan perubahan signifkan terhadap desain *database*-nya tidak perlu mengganti DBMS yang digunakan [8]. Fitur lain seperti *function* dan *procedure* juga akan sangat membantu pengembangan CCGtown di waktu yang akan datang.

CCGtown versi awal sejatinya hanya membutuhkan tiga tabel saja yaitu tabel *accounts* untuk menyimpan pengguna yang terdaftar, tabel *projects* untuk menyimpan proyek-proyek yang sudah dibuat, dan tabel *sentences* untuk menyimpan kalimat-kalimat yang akan dianotasikan. Tiga tabel tersebut sudah cukup untuk membangun *proof-of-concept* dari alat anotasi CCG yang akan dibangun. Adapun ERD[18]-nya dapat dilihat pada Gambar3.

Masing-masing tabel memiliki dua *key* yaitu ID dan $UUID$[19]. ID merupakan *primary key integer*

[14] `https://pypi.org/`
[15] `http://www.nltk.org/`
[16] Database Management System
[17] JavaScript Object Notation
[18] Entity Relationship Diagram
[19] Universally Unique IDentifier

dengan *auto increment* yang berfungsi sebagai *identifier* untuk melakukan operasi *update* maupun *delete*. Adapun $UUID$ merupakan *indexed column* yang berfungsi sebagai *indentifier* publik (dapat dilihat oleh pengguna melalui URL) yang mana digunakan untuk operasi *read*. ID tidak digunakan sebagai *identifier* publik karena pengguna dapat melakukan *brute-force* untuk mencari proyek ataupun kalimat berdasarkan ID yang bukan miliknya. Demikian itu alasan ditambahkannya atribut $UUID$. Alasan kenapa CCGtown tetap menyimpan kolom ID adalah karena ID nantinya akan digunakan untuk membuat *pagination*.

Pada tabel *accounts*, selain ID dan $UUID$ juga memiliki atribut *email* dan *password*. Masing-masing atribut tersebut menggunakan tipe data *string* atau VARCHAR di PostgreSQL. Tabel *accounts* memiliki hubungan *one-to-many* terhadap tabel *projects*. Adapun atribut tabel *projects* adalah *author_id*, *name*, *status*, dan *rules*. Atribut *author_id* merupakan *foreign key* (*indexed*) yang mengarah kepada tabel *accounts* dan tipe data yang digunakan sama dengan atribut ID yang terdapat di tabel *accounts*. Atribut *name* menggunakan tipe data *string* (VARCHAR). Atribut *status* menggunakan tipe data *integer* yang berperan sebagai *enum* (0 = *just created*, 1 = *in progress*, 2 = *finished*, dan 3 = *dropped*). Tabel *projects* memiliki hubungan *one-to-many* terhadap tabel *sentences*. Adapun atribut tabel *sentences* adalah *project_id*, *words*, *categories*, dan *derivations*. Atribut *project_id* merupakan *foreign key* (*indexed*) yang mengarah kepada tabel *projects* dan tipe data yang digunakan sama dengan atribut ID yang terdapat di tabel *projects*. Sisanya, atribut *words*, *categories*, dan *derivations* menggunakan tipe data JSON.

Desain Sistem

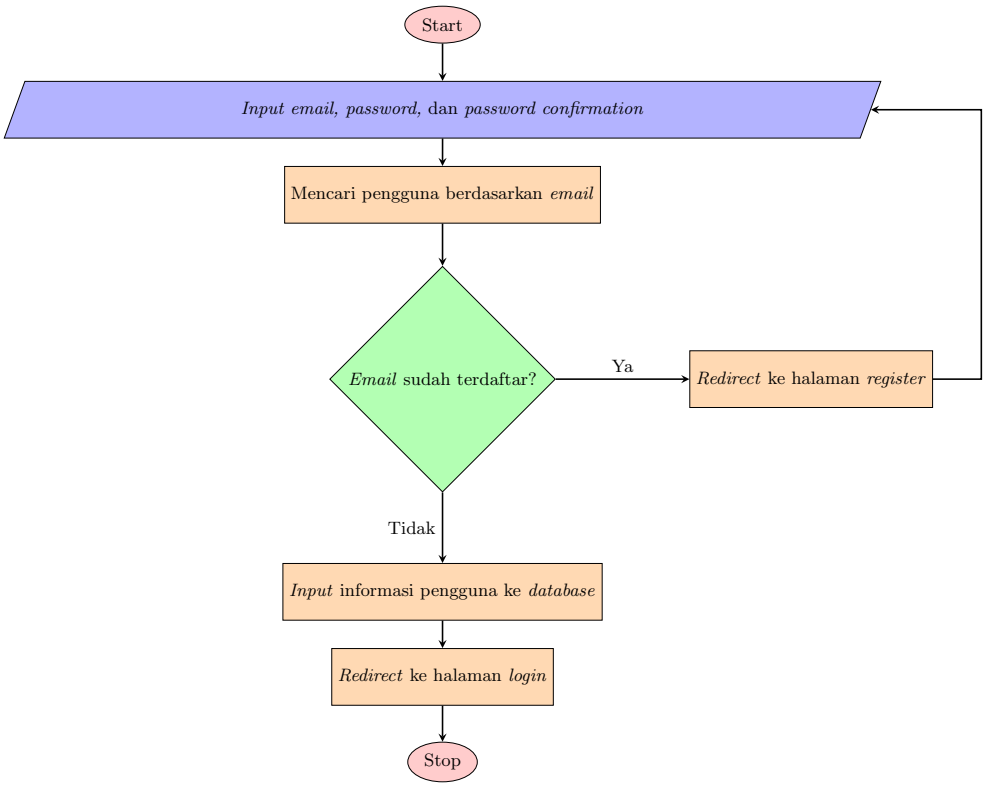
CCGtown sejatinya memiliki desain sistem yang cukup sederhana. Fungsionalitas yang akan didukung untuk versi awal adalah (1) *register* dan *login*, (2) manajemen proyek (CRUD[20]), (3) dan manajemen kalimat (CRUD). Pada manajemen kalimat, CCGtown menggunakan JavaScript untuk membuat pembuatan maupun perubahan CCG *derivation* menjadi lebih interaktif. Selain tiga fungsionalitas tersebut, CCGtown juga menambahkan fungsionalitas tambahan seperti *auto-assign category* yang dilakukan di sisi *frontend*. Kemudian, CCGtown juga menambahkan fungsionalitas tambahan di sisi *backend* yaitu CCG *derivation generator* dengan memanfaatkan *library* NLTK[1] dan kemampuan untuk melakukan *export* CCG *derivation* yang disimpan di *database*.

Pengguna harus terdaftar terlebih dahulu sebelum dapat melaukan anotasi sehingga langkah awal yang harus dibangun adalah fungsionalitas *register*. Alur proses pendaftaran pengguna dapat dilihat pada Gambar 4. Berhubung fokus saat ini adalah *proof-of-concept*, informasi yang dibutuhkan untuk mendaftar hanyalah *email* dan *password*. Adapun *password confirmation* digunakan untuk memvalidasi *password* sehingga dapat mengurangi risiko pengguna melupakan *password*-nya yang baru saja di-*input*. Saat pengguna melakukan pendaftaran, sistem akan memeriksa apakah *email* yang didaftar sudah terdapat di *database*. Apabila sudah terdaftar, pengguna akan dialihkan ke halaman *register* kembali dan mendapatkan *flash message* dengan keterangan "email sudah terdaftar". Sebaliknya, sistem akan melakukan *input* data tersebut ke dalam *database* lalu mengalihkan pengguna ke halaman *login*. Ketika dialihkan ke halaman *login*, pengguna akan melihat *flash message* dengan keterangan "pengguna berhasil didaftarkan". Pada tahap ini pengguna sudah dapat melakukan *login* ke dalam sistem CCGtown.

Pada proses "*input* informasi pengguna ke *database*" CCGtown melakukan *password hashing* dengan menggunakan Bcrypt. Informasi sensitif seperti *password* sebaiknya tidak disimpan sebagai *plain text*. Demikian itu CCGtown menggunakan *password hashing*. Apabila hal buruk terjadi seperti misalnya *data breach* (kebocoran data), *password* pengguna tidak dapat langsung digunakan. Peretas perlu mencari cara untuk memecahkan *password* tersebut. Bcrypt merupakan skema *password hashing* berbasis Blowfish *block cipher* yang didesain untuk lebih *resistant* terhadap serangan *brute-force* [5]. Serangan *brute-force* merupakan upaya peretas untuk menebak *password* dengan cara membuat *wordlist* yang kemudian dicocokkan dengan *hash* yang terbentuk satu-demi-satu. Meskipun terjadi *data breach*, peretas perlu usaha ekstra untuk dapat menebak *password* dari satu pengguna. Hal ini mengurangi kerugian yang akan dialami oleh CCGtown apabila *data breach* benar-benar terjadi.

Selanjutnya, setelah melakukan registrasi, pengguna dapat melakukan *login* ke sistem CCGtown. Proses yang dilakukan pada umumnya sama dengan aplikasi web yang memiliki kemampuan *register* dan *login*. Alur proses *login* dapat dilihat pada Gambar 5. Setelah pengguna melakukan *input email* dan *password*-nya, CCGtown akan melakukan pencarian di *database* apakah *email* yang diberikan terdaftar. Apabila tidak terdaftar, pengguna akan dialihkan ke halaman *login* dan diberikan *flash message* "*Email* dan/atau *password* tidak cocok". Pesan ini diberikan agar peretas tidak dapat mencari tahu *email* mana saja yang sudah terdaftar. Selanjutnya, apabila akun dengan *email* tersebut ada, maka langkah selanjutnya adalah mencocokkan *password* yang diberikan oleh pengguna dan *password* yang telah disimpan

[20]Create, Read, Update, Delete



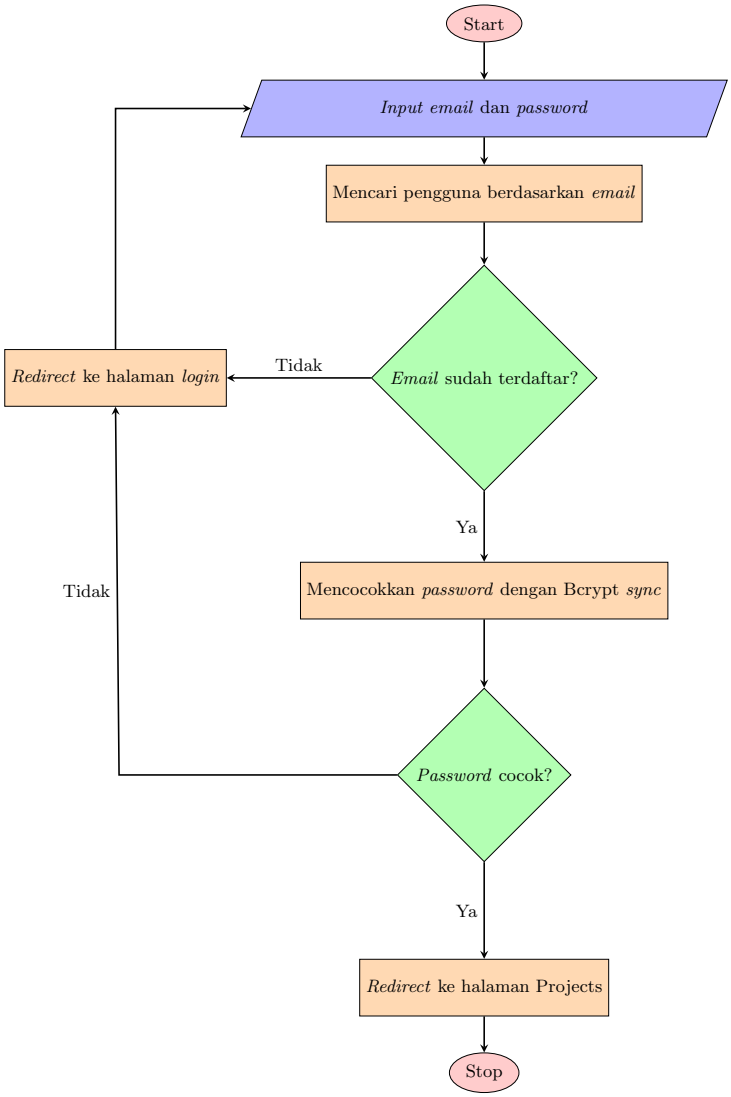
Gambar 4. Alur proses pendaftaran pengguna.

di *database*. Kemudian, sistem melakukan Bcrypt *sync*. Apabila tidak berhasil, pengguna akan dialihkan ke halaman *login* dan diberikan *flash message* "*Email* dan/atau *password* tidak cocok". Sebaliknya, pengguna akan dialihkan ke halaman Projects yang berisi daftar proyek yang telah dibuat sebelumnya.

Pada halaman Projects, pengguna dapat membuat proyek atau menghapus proyek. Tidak ada fungsionalitas spesial di halaman Projects selain CRUD pada umumnya. Satu pengguna dapat membuat banyak proyek. Tidak ada larangan tertentu terhadap penamaan proyek. Namun, sangat disarankan memberikan nama proyek yang deskriptif seperti misalnya "*Wide-range Indonesian Dataset*". Setiap proyek memiliki status yang berbeda-beda. Proyek yang baru saja dibuat akan memiliki status *just created*. Hal ini untuk memudahkan *annotator* mencari proyek mana yang baru akan dikerjakan, proyek mana yang sedang dikerjakan, proyek mana yang sudah selesai dikerjakan, atau proyek mana yang tidak jadi dikerjakan. Proyek yang telah dibuat dapat disunting maupun dihapus. Proyek yang dihapus tidak dapat dikembalikan (*undo*). Adapun penyuntingan proyek terjadi di halaman Editor.

Pada halaman Editor, pengguna dapat menyunting informasi proyek seperti nama proyek, status proyek, dan *rules* yang akan digunakan untuk melakukan *generate* CCG *derivation* via NTLK. Selain itu, pengguna juga dapat menambahkan kalimat baru yang akan dianotasi. Pengguna dapat menambahkan lebih dari satu kalimat sekaligus. Kalimat-kalimat tersebut akan di-*tokenize* menggunakan *library* NLTK. Ekstensi yang digunakan untuk proses *tokenize* ini adalah punkt. Setelah itu, barulah pengguna dapat melakukan penganotasian terhadap kalimat-kalimat yang telah ditambahkan. Terdapat dua cara untuk memberikan anotasi yaitu secara langsung di halaman Editor atau dapat juga dilakukan di Editable CCG Modal. Saat ini CCGtown belum mendukung penganotasian terhadap *compound words*. CCGtown saat ini juga belum mendukung penganotasian CCG dengan semantik. Versi awal CCGtown hanya mendukung penganotasian CCG secara sintaksis saja.

Setelah semua kata dalam suatu kalimat diberikan anotasi, pengguna dapat melakukan *generate* CCG *derivation*. Hal ini dapat dilakukan berkat bantuan *library* NLTK. Kami mengambil sebuah *rules* dari tabel *projects* dan kemudian kami mengambil semua *words* serta *categories* dari tabel *sentences* yang merupakan bagian dari proyek tersebut. Kolom *words* merupakan kumpulan kata dari kalimat yang telah di-*tokenize*. Adapun kolom *categories* merupakan anotasi CCG *category*-nya. *Pseudocode* untuk *generate* CCG *derivation* dapat dilihat pada Kode 3.1 dengan asumsi anotasi yang diberikan absah (dapat dibuat CCG *derivation*-nya). Kode *next* tersebut akan mengambil satu dari banyak kemungkinan *derivation* yang dapat dibuat. Contoh *object* yang di-*return* dapat dilihat pada Kode 3.2. Untuk ke-



Gambar 5. Alur proses *login* ke sistem CCG.

pentingan *rendering* di sisi *frontend*, *key* seperti *from* dan *to* sangat diperlukan. *Key from* dan *key to* merepresentasikan *index* posisi terhadap *array words*. Dengan bantuan kedua *key* tersebut, *frontend* dapat melakukan kalkulasi posisi masing-masing elemen yang terdapat di *object derivations*.

Kode 3.1: Pseudocode untuk melakukan *generate* CCG *derivation*.

```
from nltk.ccg import chart, lexicon

def generateCCGDerivation(rules, words, categories, target_words):
    lex = rules + '\n\n'
    for i in range(len(words)):
        lex += words[i] + ' => ' + categories[i] + '\n'

    lex = lexicon.parseLexicon(lex)
    parser = chart.CCGChartParser(lex, chart.DefaultRuleSet)
    result = next(parser.parse(target_words))
    derivations = makeCCGDeriv(result)

    return derivations
```

Kode 3.2 didapatkan dari fungsi *makeCCGDeriv* yang terdapat pada Kode 3.1. Fungsi *makeCCGDeriv* sederhananya mengambil *Tree* yang didapatkan dari *parser.parse* kemudian melakukan *tree traversal*. Semua *leaf*, diambil dari paling "kiri", diletakkan di elemen pertama *derivations*. Selanjutnya, kita berjalan melalui *parent* dari *leaf* tersebut hingga ke *root* mencari bentuk CCG *derivation*-nya. Banyaknya baris yang dibutuhkan oleh CCG *derivation* dapat dilihat dari *height* yang dimiliki oleh *Tree* tersebut. Kemudian, hasil dari CCG *derivation* (umumnya berupa S) merupakan elemen terakhir *derivations*.

Kode 3.2: Contoh *derivations object* yang di-*return*.

```
[
  [
    { "to": 0, "from": 0, "word": "You" },
    { "to": 1, "from": 1, "word": "prefer" },
    { "to": 2, "from": 2, "word": "that" },
    { "to": 3, "from": 3, "word": "cake" }
  ],
  [
    { "to": 0, "from": 0, "category": "NP" },
    { "to": 1, "from": 1, "category": "((S\NP)/NP)" },
    { "to": 2, "from": 2, "category": "(NP/N)" },
    { "to": 3, "from": 3, "category": "N" }
  ],
  [
    { "to": 3, "from": 2, "category": "NP", "operator": ">" }
  ],
  [
    { "to": 3, "from": 1, "category": "(S\NP)", "operator": ">" }
  ],
  [
    { "to": 3, "from": 0, "category": "S", "operator": "<" }
  ]
]
```

Selain memiliki kemampuan untuk melakukan *generate* CCG *derivation*, CCGtown juga memiliki kemampuan untuk melakukan *auto-assign* CCG *category*. Token kata yang sudah dianotasi oleh pengguna akan disimpan ke dalam suatu *dictionary*. Untuk setiap kata yang belum dianotasi, CCGtown akan memeriksa apakah token kata tersebut sebelumnya sudah dianotasi. Apabila sudah, CCGtown akan memberikan anotasi secara otomatis. Suatu token kata mungkin memiliki lebih dari satu anotasi. CCGtown hanya akan mengambil satu anotasi saja. Akibatnya, pengguna sebaiknya tetap melakukan peninjauan. Kendati demikian, setidaknya kegiatan anotasi yang repetitif dapat berkurang sehingga memudahkan dan mempercepat proses anotasi.

Deployment

CCGtown menggunakan Heroku sebagai *cloud application platform* untuk *hosting* aplikasi tersebut agar dapat diakses oleh pengguna via internet. Heroku memberikan kemudahan untuk melakukan *deployment* dan *scaling* apabila dibutuhkan [6]. Selain itu, integrasi Heroku dengan GitHub memberikan fitur *automatic deployment* ke *branch* tertentu sehingga pengembang tidak perlu melakukan instalasi manual secara repetitif. Demikian itu, pengembang tidak akan direpotkan dengan proses *deployment*. Apa yang perlu dilakukan adalah melakukan *setup* terlebih dahulu kemudian *automatic deployment* dapat berjalan sampai dengan dimatikan kembali.

CCGtown menggunakan *web framework* yang proses *setup* dan *run*-nya dapat dijalankan hanya dengan beberapa perintah CLI[21] saja. Demikian itu, Heroku dapat menjalankan perintah-perintah tersebut

[21] command line interface

dan ketika semua perintahnya memberikan *return value* berupa 0 (sukses), secara otomatis Heroku akan menimpa aplikasi yang sedang berjalan dengan yang baru. *Dependency* NLTK seperti *punkt* juga tidak perlu di-*install* manual. Hanya dengan menambahkan berkas "nltk.txt", Heroku akan secara otomatis melakukan instalasi semua *dependency* NLTK yang terdaftar di berkas tersebut. Adapun *dependency* untuk bahasa Python (PyPI) yang digunakan tersedia di dalam berkas "requirements.txt" dan Heroku juga akan melakukan instalasi semua *dependency* yang terdaftar di dalam berkas tersebut.

4. Evaluasi

4.1 Hasil Pengujian

Pengujian dilakukan dengan menggunakan *state transition testing* yang mana merupakan bagian dari *black-box testing*. Dalam hal ini, apa yang akan diuji adalah fungsionalitas dari CCGtown dan *testing* ini digunakan untuk mencari apakah ada *bug* dari sistem yang telah dibangun. Aturan *state transition testing* cukup sederhana yaitu (1) kunjungi setiap *state*, (2) cakup semua transisi, dan (3) pastikan tidak ada *state* yang tak dapat dijangkau atau tak dapat dikembalikan[2]. Untuk mempercepat proses pengujian, kami menggunakan *state transition table* sebagai ganti dari *state transition diagram*. Daftar *state* dapat dilihat pada Tabel 1. Pada Tabel 1, S1 dan S2 merupakan *initial state* dan S5 merupakan *final state*.

Pengujian dikelompokkan berdasarkan antarmuka yang dilihat oleh pengguna. Umumnya berupa halaman web atau modal layar penuh. Masing-masing tabel memiliki kolom nomor, *start state*, *input*, *output*, dan *finish state*. Adapun transisi yang telah dikumpulkan dapat dilihat pada Tabel 2 untuk halaman registrasi, Tabel 3 untuk halaman login, Tabel 4 untuk halaman proyek, Tabel 5 untuk halaman editor, Tabel 6 untuk Editable CCG Derivation Modal, dan Tabel 7 untuk Configure Derivation Modal.

4.2 Analisis Hasil Pengujian

Pengguna dapat melakukan login secara langsung apabila telah memiliki akun sehingga dalam hal ini *state* S2 merupakan *start state*. Sebaliknya, apabila belum memiliki akun, pengguna dapat melakukan registrasi terlebih dahulu. Artinya, *state* S1 juga merupakan *start state*. Tujuan CCGtown adalah untuk memfasilitasi *annotator* dalam memberikan anotasi CCG. Demikian itu, *final state* CCGtown terdapat pada fitur *export to* JSON yang mana merupakan *state* S5.

Pada Tabel 2, pengguna akan diarahkan ke *state* selanjutnya yaitu S2 ketika *email* yang didaftarkan belum terdaftar dan *password* yang diberikan absah saja. Selain itu, pengguna akan dikembalikan ke *state* S1 karena aksi yang dilakukan sebelumnya tidak berhasil dilakukan. Adapun pada Tabel 3, penggunaan akan diarahkan ke halaman proyek yang mana merupakan *state* S3 ketika *email* terdaftar dan *password* yang diberikan sesuai dengan yang didaftarkan sebelumnya. Selain itu, pengguna akan diarahkan kembali ke halaman login.

Pada Tabel 4, ada cukup banyak hal yang dapat dilakukan oleh penggunaan. Mulai dari *logout* dari CCGtown, menambahkan proyek baru, menghapus suatu proyek, hingga melakukan *export to* JSON. Selanjutnya, apabila pengguna memutuskan untuk melihat atau meng-*edit* suatu proyek, ia akan diarahkan ke halaman editor yang mana merupakan *state* S4. Pada Tabel 5, hanya ada dua transisi yang memiliki *finish state* selain S4 yaitu pada nomor 1, 11, dan 14. Selain ketiga transisi tersebut, pengguna tetap berada di *state* S4 yaitu halaman editor. Pengguna juga dapat langsung menuju *final state* dengan mengklik *export to JSON* pada menu yang sudah tersedia. Adapun transisi yang lain adalah aksi *logout* dan aksi *view ccg derivation* yang mana akan membuka Editable CCG Derivation Modal (*state* S6).

Pada Tabel 6, pengguna dapat mengubah anotasi CCG dan dapat mengubah penurunan CCG. Selain itu, pengguna dapat keluar dari modal tersebut atau dapat mengatur penurunan yang ada. Pengaturan dari penurunan tersebut dapat dilakukan di Configure Derivation Modal yang mana merupakan *state* S7. Pada Tabel 7, pengguna dapat menambah atau menghapus konfigurasi penurunan CCG. Transisi lainnya adalah keluar dari modal tersebut atau simpan konfigurasinya. Demikian itu, dilihat dari *state transition table* yang ada, CCGtown selalu dapat kembali ke *state* sebelumnya.

5. Kesimpulan

Berdasarkan *state transition testing*, transisi yang terjadi pada CCGtown dapat berjalan maju (misal: dari S3 ke S4) atau pun dapat berjalan mundur (misal: dari S4 ke S2). Hal tersebut memungkinkan pengguna untuk kembali ke *initial state* dan berakhir di *final state*. Satu-satunya *state* yang tidak memiliki transisi baik maju maupun mundur adalah S7 yaitu *export to* JSON. Hal ini dikarenakan

Tabel 1. **Pemetaan State**

State	Keterangan
S1	Halaman registrasi pengguna
S2	Halaman login
S3	Halaman proyek
S4	Halaman editor
S5	Halaman export JSON
S6	Editable CCG Derivation Modal
S7	Configure Derivation Modal

ketika pengguna mengklik tombol *export*, tab baru di *browser* pengguna akan terbuka sehingga akan ada setidaknya dua tab aplikasi CCGtown yang terbuka. Demikian itu, ketika pengguna menutup halaman hasil *export* tersebut lalu membuka CCGtown kembali, secara otomatis *state* aktif tidak lagi S7. Adapun *state*-nya dapat berupa S3 atau S4. Apabila sesi pengguna kadaluwarsa, *state* yang akan aktif adalah S2. Demikian itu, berdasarkan pengujian ini semua fungsionalitas CCGtown sudah berjalan dengan baik dan dapat digunakan oleh pengguna tanpa masalah.

Daftar Pustaka

[1] S. Bird, E. Klein, and E. Loper. *Natural Language Processing with Python*. O'Reilly Media, Inc., 1st edition, 2009.

[2] R. Black. *Pragmatic Software Testing: Becoming an Effective and Efficient Test Professional*. Wiley, 2016.

[3] K. Evang, L. Abzianidze, and J. Bos. CCGweb: a new annotation tool and a first quadrilingual CCG treebank. In *Proceedings of the 13th Linguistic Annotation Workshop*, pages 37–42, Florence, Italy, Aug. 2019. Association for Computational Linguistics.

[4] J. Lambek. *Categorial and Categorical Grammars*, pages 297–317. Springer Netherlands, Dordrecht, 1988.

[5] K. Malvoni, S. Designer, and J. Knezovic. Are your passwords safe: Energy-efficient bcrypt cracking with low-cost parallel hardware. 08 2014.

[6] N. Middleton and R. Schneeman. *Heroku: Up and Running: Effortless Application Deployment and Scaling*. O'Reilly Media, 2013.

[7] R. Rojas. A tutorial introduction to the lambda calculus. *CoRR*, abs/1503.09060, 2015.

[8] H. Schönig. *Mastering PostgreSQL 10: Expert techniques on PostgreSQL 10 development and administration*. Packt Publishing, 2018.

[9] M. Steedman. Categorial grammar. Technical report, 1992.

[10] M. Steedman. A very short introduction to ccg. Technical report, 1996.

Lampiran

Tabel 2. **State transition pada halaman registrasi pengguna**

No.	Start State	Input	Output	Finish State
1	S1	invalid email atau invalid password	diarahkan ke halaman registrasi	S1
2	S1	email terdaftar	diarahkan ke halaman registrasi	S1
3	S1	email tidak terdaftar dengan invalid password	diarahkan ke halaman registrasi	S1
4	S1	email tidak terdaftar dengan valid password	diarahkan ke halaman login	S2

Tabel 3. **State transition pada halaman login**

No.	Start State	Input	Output	Finish State
1	S2	email terdaftar dan password sesuai	diarahkan ke halaman proyek	S3
2	S2	email tidak terdaftar atau password salah	diarahkan ke halaman login	S2

Tabel 4. **State transition pada halaman proyek**

No.	Start State	Input	Output	Finish State
1	S3	mengklik ikon profil kemudian logout	diarahkan ke halaman login	S2
2	S3	mengklik nama proyek atau edit proyek	diarahkan ke halaman editor	S4
3	S3	mengklik "remove project" kemudian "cancel"	tidak terjadi apa-apa	S3
4	S3	mengklik "remove project" kemudian "remove"	diarahkan ke halaman proyek dan proyek yang dipilih terhapus	S3
5	S3	mengklik "create new project" kemudian "cancel"	tidak terjadi apa-apa	S3
6	S3	mengklik "create new project" kemudian "create project" dengan memasukkan nama proyek	diarahkan ke halaman proyek dan proyek baru dibuat	S3
7	S3	mengklik "export to JSON"	tab baru terbuka dan menampilkan hasil export	S5
8	S3	sesi pengguna kadaluwarsa atau pengguna belum login	diarahkan ke halaman login	S2

Tabel 5. **State transition pada halaman editor**

No.	Start State	Input	Output	Finish State
1	S4	mengklik ikon profil kemudian logout	diarahkan ke halaman login	S2
2	S4	melakukan suatu perubahan kemudian mengklik "save"	diarahkan ke halaman editor dan semua perubahan disimpan	S4
3	S4	mengubah/menghapus anotasi di suatu kata	anotasi pada kata tersebut berubah	S4
4	S4	mengklik "add new sentences" kemudian klik cancel	tidak terjadi apa-apa	S4
5	S4	mengklik "add new sentences" kemudian klik "add sentences" setelah menuliskan kalimat/paragraf	kalimat-kalimat baru ditambahkan	S4
6	S4	mengklik "generate ccg derivation" kemudian klik "cancel"	tidak terjadi apa-apa	S4
7	S4	mengklik "generate ccg derivation" kemudian klik "generate now" dengan invalid lexicon	ccg derivation tidak di-generate	S4
8	S4	mengklik "generate ccg derivation" kemudian klik "generate now" dengan valid lexicon	ccg derivation di-generate	S4
9	S4	mengklik "remove sentence" kemudian klik "cancel"	tidak terjadi apa-apa	S4
10	S4	mengklik "remove sentence" kemudian klik "remove sentence"	kalimat tersebut dihapus	S4
11	S4	mengklik "view ccg derivation"	editable ccg modal terbuka	S6
12	S4	mengklik "remove project" kemudian klik "cancel"	tidak terjadi apa-apa	S4
13	S4	mengklik "remove project" kemudian klik "remove project"	proyek dihapus lalu diarahkan ke halaman proyek	S3
14	S4	mengklik "export to JSON"	tab baru terbuka dan menampilkan hasil export	S5
15	S4	mengklik "auto-assign category" kemudian klik "cancel"	tidak terjadi apa-apa	S4
16	S4	mengklik "auto-assign category" kemudian klik "auto-assign now"	kata yang belum dianotasi diberikan anotasi dari kata yang sama di kalimat lain	S4
17	S4	sesi pengguna kadaluwarsa atau pengguna belum login	diarahkan ke halaman login	S2

Tabel 6. **State transition pada Editable CCG Derivation Modal**

No.	Start State	Input	Output	Finish State
1	S6	mengklik tombol "X"	editable ccg derivation modal ditutup	S4
2	S6	mengklik "add new row"	baris kosong baru ditambahkan	S6
3	S6	mengklik anotasi kemudian mengubah/menghapus anotasi	anotasi yang dipilih berubah	S6
4	S6	mengklik "configure derivation on this row"	configure derivation modal terbuka	S7

Tabel 7. **State transition pada Configure CCG Modal**

No.	Start State	Input	Output	Finish State
1	S7	mengklik tombol "X" atau "cancel	configure ccg modal ditutup	S6
2	S7	mengklik tombol "save"	perubahan disimpan kemudian modal ditutup	S6
3	S7	mengklik "add entry"	konfigurasi kosong baru ditambahkan	S7
4	S7	mengklik tombol dengan ikon tempat sampah	konfigurasi yang dipilih terhapus	S7