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

Tugas Akhir
diajukan untuk memenuhi salah satu syarat
memperoleh gelar sarjana
dari Program Studi Informatika
Fakultas Informatika
Universitas Telkom

1301160479
Wisnu Adi Nurcahyo

Program Studi Sarjana Informatika
Fakultas Informatika
Universitas Telkom
Bandung

2021

LEMBAR PENGESAHAN

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

CCGtown: A Web-based Semi-automatic Combinatory Categorial Grammar (CCG) Annotation Tool

NIM: 1301160479

Wisnu Adi Nurcahyo

Tugas akhir ini telah diterima dan disahkan untuk memenuhi sebagian syarat memperoleh
gelar pada Program Studi Sarjana Informatika
Fakultas Informatika
Universitas Telkom

Bandung, 20 Januari 2021
Menyetujui

Pembimbing I

Dr. Ade Romadhony, S.T., M.T.
NIP: 06840042

Ketua Program Studi
Sarjana Informatika,

Niken Dwi Wahyu Cahyani, S.T., M.Kom. PhD

NIP: 00750052

LEMBAR PERNYATAAN

Dengan ini saya, Wisnu Adi Nurcahyo, menyatakan sesungguhnya bahwa Tugas Akhir saya dengan judul "**CCGtown: Alat Anotasi Combinatory Categorial Grammar (CCG) Semi-otomatis Berbasis Web**" beserta dengan seluruh isinya adalah merupakan hasil karya sendiri, dan saya tidak melakukan penjiplakan yang tidak sesuai dengan etika keilmuan yang belaku dalam masyarakat keilmuan. Saya siap menanggung resiko/sanksi yang diberikan jika dikemudian hari ditemukan pelanggaran terhadap etika keilmuan dalam buku TA atau jika ada klaim dari pihak lain terhadap keaslian karya.

Bandung, 20 Januari 2021

Yang Menyatakan,

Wisnu Adi Nurcahyo

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

Abstrak

Dokumen ini merupakan panduan penulisan jurnal Tugas Akhir (TA) di lingkungan Fakultas Informatika Universitas Telkom. Meskipun demikian, dimungkinan/dipersilahkan untuk pembimbing TA menggunakan struktur penulisan yang tidak sama persis dengan yang ada di dokumen ini. Panjang abstrak tidak lebih dari 200 kata dan diketik dalam ukuran huruf 10 pts. TA sebagai salah satu sarana latihan penulisan akademik dan memperjelas tulisan, abstrak dibagi menjadi empat paragraf atau sub-bagian. Setiap sub bagian bisa diberi judul yang digaris bawahi. Abstrak berisi apa, mengapa, bagaimana, dan hasil utama (kesimpulan).

***Apa permasalahan pada topik*. Yang juga menjelaskan latar belakang permasalahan topik. Sebaiknya tuliskan juga apa masukan dan keluaran secara sangat singkat.**

***Mengapa topik menarik atau penting*. Sebisa mungkin tuliskan contohnya secara sangat singkat. Pada bagian ini sebaiknya ditulis juga *apa masalah/kekurangan yang terjadi unt kondisi saat ini* (gap antara kondisi sekarang dengan yang diharapkan)?**

***Bagaimana solusinya*. Jelaskan secara garis besar sistem solusi yang telah dilakukan. Biasanya penjelasan solusi ini merupakan yang terpanjang pada abstrak.**

***Hasil utama*. Hasil utama dari eksperimen ditulis singkat dua-tiga kalimat. Akan lebih baik (optional), kalau dituliskan secara eksplisit kontribusi yang telah dihasilkan. Kontribusi bisa dituliskan diantara bagian solusi dan hasil eksperimen.**

Pastikan abstrak pada jurnal TA tidak copas dari abstrak proposal TA. Pada abstrak proposal kadang ada kata *akan*, seperti misalnya *yang akan dilakukan*; sedangkan pada abstrak Jurnal TA tidak ada kata *akan* spt itu. Tidak boleh ada sitasi pada abstrak. Pada abstrak tidak menggunakan penamaan, simbol atau istilah yang teknis, misalnya *minsup* untuk menyatakan nilai support minimal.

Kata kunci : merupakan kata-kata kunci yang menjelaskan isi tulisan, biasanya bisa diambil dari judul dan abstrak. Maksimal enam buah dan ditulis dengan huruf kecil, kecuali singkatan

Abstract

The abstract should state briefly the general aspects of the subject and the main concolusions. The length of abstract should bo no more than 200 word and should be typed be with 10 pts.

Keywords: keyword should be chosen that they best describe the contents of the paper and should be typed in lower-case, except abbreviation. Keyword should be no more than 6 word

1. Pendahuluan

Latar Belakang

Riset pemrosesan bahasa alami untuk bahasa Indonesia saat ini masih terbilang sedikit. Bahkan, masih banyak area riset yang belum tersentuh seperti contohnya *combinatory categorial grammar* (CCG). Sementara itu, riset mengenai CCG untuk bahasa Inggris sudah cukup matang. Adapun untuk bahasa lainnya (seperti bahasa Vietnam) sudah mulai menggunakan CCG di dalam penelitiannya (**?**). Agar dapat menerapkan CCG di dalam aplikasi yang dibangun, *tools* seperti CCG *parser* dan CCG *supertagger* harus tersedia terlebih dahulu. Masing-masing dari *tools* tersebut memerlukan *dataset* agar dapat memberikan hasil yang akurat.

Umumnya terdapat dua cara yang paling sering digunakan untuk mengembangkan CCG *supertagger* maupun CCG *parser* bahasa lokal yaitu (1) membangun *dataset* CCG *supertag* secara manual maupun semi-otomatis atau (2) melakukan transfer *dataset* dari CCGbank (atau dari sumber lainnya) ke dalam bahasa lokal dengan cara melakukan alih bahasa dan bila perlu melakukan penyesuaian untuk *supertag*-nya (**?**). Proses pembangunan *dataset* umumnya menggunakan bantuan *annotation tool* agar proses anotasinya menjadi lebih mudah. Salah satu *annotation tool* yang dapat digunakan adalah CCGweb (**?**).

Tugas akhir ini berusaha untuk membangun alat anotasi CCG baru dengan UI/UX yang lebih baik dari CCGweb. Selain itu, dengan bantuan NLTK alat anotasi ini dapat melakukan *generate* untuk CCG *derivation*-nya kemudian pengguna juga dapat mengubah *derivation*-nya apabila diperlukan. Tujuan dari dibangunnya alat anotasi CCG ini adalah untuk mempermudah proses anotasi yang repetitif. Selanjutnya, *dataset* CCG pertama untuk bahasa Indonesia diharapkan dapat dipublikasikan.

Topik dan Batasannya

Sub-bagian ini bisa juga dinamakan Perumusan Masalah atau Identifikasi Masalah. Untuk nama dalam Bahasa Inggris nama yang populer adalah *Problem Statement* atau *Problem Identification.*

Sub-bagian ini mempunyai fungsi sebagai penjelasan tentang topik TA yaitu apa isu/permasalahan yang akan dikerjakan. Untuk lebih memperjelas bisa juga disampaikan definisi atau pengertian. Penyampaian definisi dan penjelasan pada sub-bagian ini sebaiknya dilakukan dalam tulisan naratif dan informal (tanpa formula matematis) apa topik permasalahan yang telah dikerjakan untuk TA. Untuk mempermudah dalam menuliskan sub-bagian ini, dapat dipandang membuat penjelasan kata-kata kunci (pada abstrak) dan judul TA. Dengan penjelasan di sub-bagian ini, maka topiknya menjadi jelas bagi pembaca. Kalau digambarkan dalam sebuah algoritma, maka salah satu materi utama pada sub-bagian ini menjelaskan apa input dan output dari algoritma tersebut. Oleh karena itu, sangat dianjurkan untuk menerangkan apa input dan output, serta sebuah contoh kasusnya secara sangat singkat.

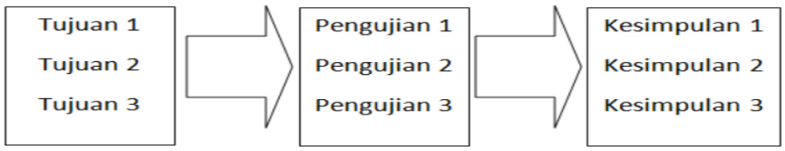
Sebutkan batasan pekerjaan yang ada. Batasan adalah kondisi-kondisi penyederhaan permasalahan, sehingga membuat pekerjaan semakin jauh dari ideal. Batasan masalah berisi pembatasan-pembatasan permasalahan agar menjadi lebih sederhana sehingga bisa/layak dikerjakan sebagai TA yang empat SKS dalam satu semester. Batasan diperlukan karena keterbatasan sumber daya saat pengerjaa TA, misalnya keterbatasan waktu pengerjaan yang hanya satu semester, keterbatasan data pendukung (misalnya tidak tersedianya korpus pengetahuan yang diperlukan) dan keterbatas kemampuan (misalnya untuk implementasi algoritma yang kompleks, dalam implementasinya diimplementasikan bentuk penyederhanaan). Salah satu ciri batasan yang bisa dipakai adalah bila bisa digunakan pada sub-bagian Saran (pada bagian Kesimpulan) agar TA berikutnya melonggarkan atau meniadakan batasan tersebut. Penyederhanaan yang dituliskan untuk batasan, antara lain meliputi data yang ditangani/digunakan, misalnya jumlah data yang digunakan relatif sedikit, dan proses yang dikerjakan, misalnya ada satu subproses yang dikerjakan secara manual. Sebaiknya setiap batasan diberi alasan, misalnya jumlah data yang digunakan hanya 500 buah (relatif sedikit dibandingkan banyak penilitian unt topik sejenis) karena keterbatasan kemampuan komputer yang tersedia. Contoh lain, misalnya proses pelabelan peran semantik pada kalimat Bahasa Indonesia dilakukan secara manual, karena saat ini belum ditemukan alat bantu otomatis unt pelabelan peran semantik untuk Bahasa Indonesia yang efektif. Contoh batasan masalah yang tidak perlu misalnya sudah jelas tercerminkan pada judul.

Tujuan

Sub-bagian Tujuan ini menerangkan kondisi apa yang hendak dicapai atau pertanyaan yang hendak dicari jawabannya. Sebisa mungkin tuliskan kondisi yang hendak dicapai yang terukur (bisa diukur dengan metrik evaluasi yang ditetapkan). Penulisan diupayakan dalam bentuk narasi (bukan berupa poin-poin).

Tujuan-tujuan yang ditetapkan menjadi bahan untuk menentukan skenario eksperimen yang dilakukan. atau dengan kata lain eksperimen dilakukan sesuai dengan tujuannya. Kemudian, kesimpulan pada jurnal TA harus selaras dengan tujuan. Hal ini bisa diilustrasikan pada Gambar 1 atau Tabel 1.

Organisasi Tulisan



Gambar 1. **Keterkaitan antara tujuan, pengujian dan kesimpulan**

Tabel 1. **Keterkaitan antara tujuan, pengujian dan kesimpulan**

No	Tujuan	Pengujian	Kesimpulan
1	Tujuan 1	Pengujian 1	Kesimpulan 1
2	Tujuan 2	Pengujian 2	Kesimpulan 2
3	Tujuan 3	Pengujian 3	Kesimpulan 3

Pada sub-bagian ini dituliskan bagian-bagian selanjutnya (setelah Pendahuluan) pada jurnal TA ini, disertai penjelasan sangat singkat.

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 (**?**). 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 **??**, **??**, dan **??** dimana A, B, dan C merupakan himpunan bagian dari M **?**. 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 **??** dan **??**. Terakhir, persamaan **??** dapat diabaikan apabila dihadapkan dengan *multiplicative system* yang tidak asosiatif. Sementara itu, apabila *semigroup*-nya merupakan sebuah *monoid* dengan identitas 1 maka kita dapatkan **??** dimana $I = \{1\}$.

$$A \cdot B = \{x \cdot y \in M \mid x \in A \wedge y \in B\} \tag{1}$$
$$C/B = \{x \in M \mid \forall_{y \in B} x \cdot y \in C\} \tag{2}$$
$$A \backslash C = \{y \in M \mid \forall_{x \in A} x \cdot y \in C\} \tag{3}$$

$$A \cdot B \subseteq C \quad \text{jika dan hanya jika} \quad A \subseteq C/B \tag{4}$$
$$A \cdot B \subseteq C \quad \text{jika dan hanya jika} \quad B \subseteq A \backslash C \tag{5}$$

$$(A \cdot B) \cdot C = A \cdot (B \cdot C) \tag{6}$$
$$I \cdot A = A = A \cdot I \tag{7}$$

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

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

Gambar 2. Kamus yang memetakan token kata ke bentuk CCG *lexicon*-nya.

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 (**?**). 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 **??**. Apabila kita memiliki kalimat "Pamungkas dan Setyo menyukai rendang", maka kita dapatkan:

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

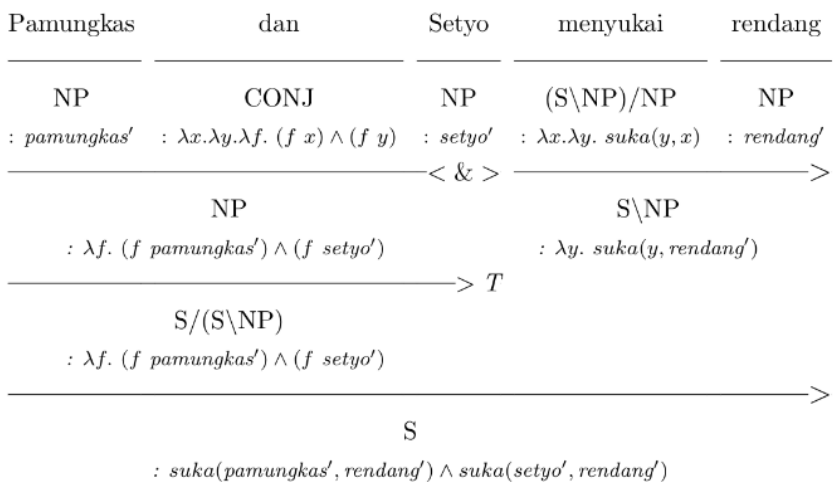
Ada beberapa operasi yang dapat dilakukan dalam CCG. *Operand* dari operasi yang dimaksud adalah *category*. Berdasarkan contoh di atas, akan ada tiga operasi yang dijalankan yaitu *coordination*, *forward application*, dan *type rising*. Untuk mendapatkan hasil yang diinginkan, kita lakukan *type rising* sebelum *forward application* di akhir. Sehingga, kita dapatkan Gambar **??**. Berdasarkan hasil evaluasi tersebut, kita dapatkan *query* **??** 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.

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

Lambda Calculus

Lambda calculus (λ-*calculus*) merupakan sebuah formalisme yang dikembangkan oleh Alonzo Church sebagai alat yang digunakan untuk memahami konsep komputasi yang efektif (**?**). 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



Gambar 3. Contoh CCG *derivation* dengan operasi *coordination*, *forward application*, dan *type rising*.

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[1] merupakan *open source graphical annotation tool* pertama untuk CCG (**?**). 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 **??**.

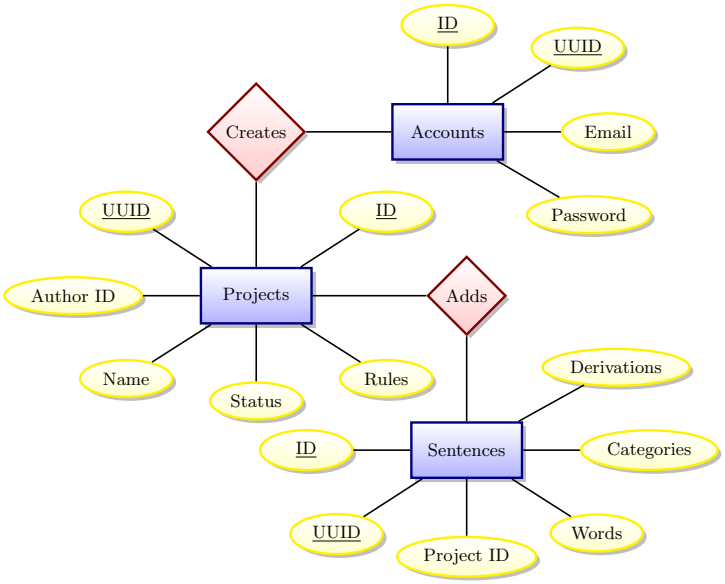
Untuk dapat menggunakan CCGweb, kita perlu melakukan instalasi terlebih dahulu. Selanjutnya barulah kita dapat menambahkan kalimat-kalimat yang ingin dianotasi. Satu instalasi CCGweb hanya dapat digunakan untuk satu proyek anotasi sehingga apabila kita memiliki lebih dari satu proyek maka kita perlu melakukan instalasi CCGweb yang baru. Demikian itu, CCGtown[2] hadir dengan fitur *multi-project* dan tanpa perlu melakukan instalasi di komputer lokal karena aplikasinya *hosted* sehingga dapat diakses kapan pun.

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 [3]. Salah satu *library* penting yang digunakan sebagai dasar dari fitur penganotasian semi-otomatis adalah NTLK [4]. 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 satu-

[1] `https://github.com/texttheater/ccgweb`
[2] `https://github.com/wisn/ccgtown`
[3] `https://pypi.org/`
[4] `http://www.nltk.org/`



Gambar 4. Conceptual Entity Relationship Diagram (ERD) CCGtown

nya adalah desain dari *database* yang akan digunakan.

Desain Database

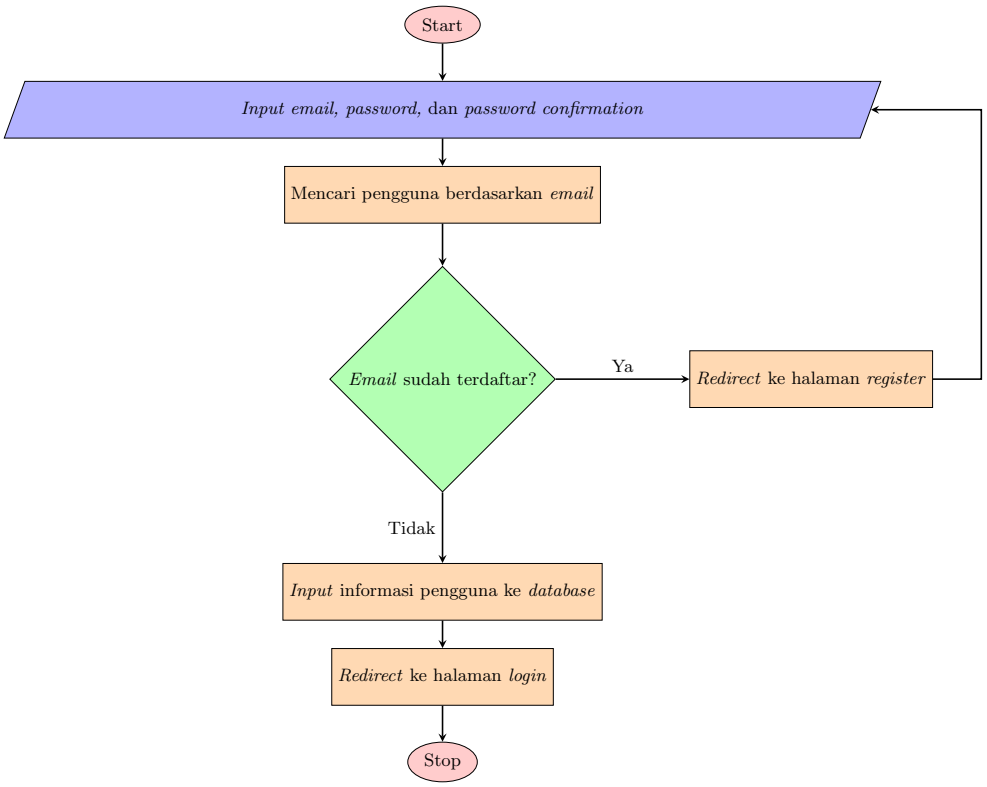
CCGtown menggunakan PostgreSQL sebagai DBMS[5]-nya. Hal ini karena PostgreSQL memiliki kemampuan untuk menyimpan struktur data JSON[6] sehingga memudahkan CCGtown untuk menyimpan format JSON dari CCG *derivation* yang telah dimanipulasi oleh pengguna melalui fitur *editable CCG derivation*. PostgreSQL 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. 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[7]-nya dapat dilihat pada Gambar**??**.

Masing-masing tabel memiliki dua *key* yaitu ID dan $UUID$[8]. ID merupakan *primary key integer* 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

[5]Database Management System
[6]JavaScript Object Notation
[7]Entity Relationship Diagram
[8]Universally Unique IDentifier



Gambar 5. Alur proses pendaftaran pengguna.

terdapat di tabel *projects*. Sisanya, atribut *words*, *categories*, dan *derivations* menggunakan tipe data JSON.

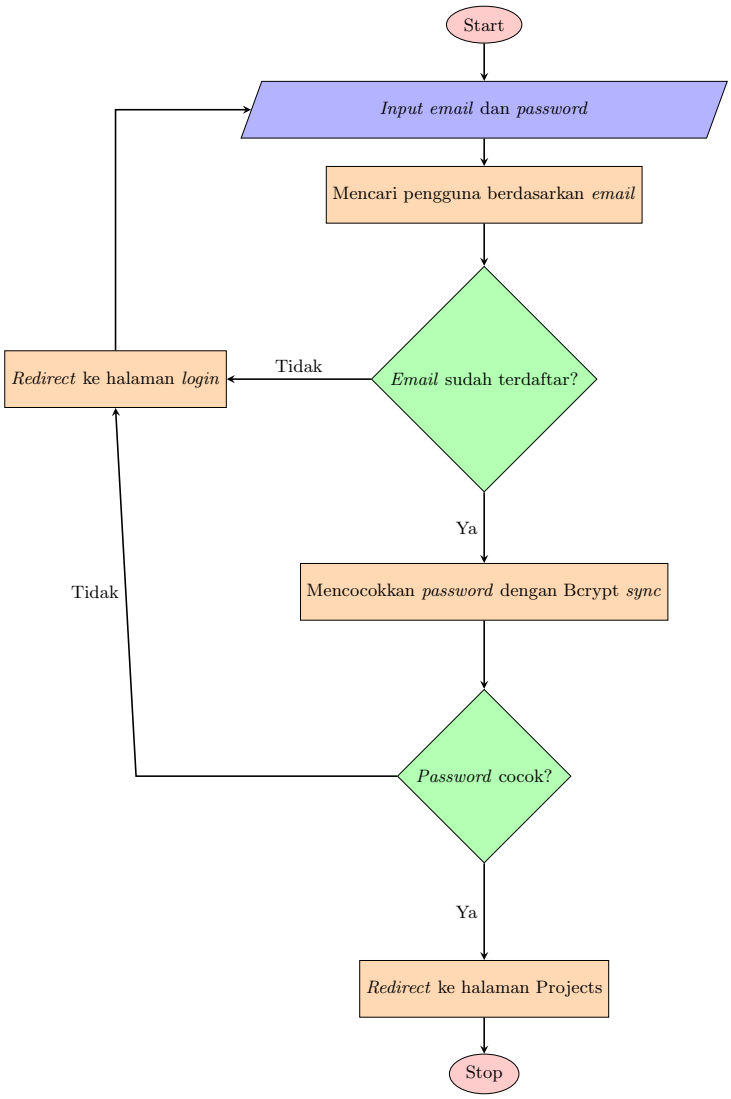
Desain Sistem

CCGtown sejatinya memiliki desain sistem yang cukup sederhana. Fungsionalitas yang akan didukung untuk versi awal adalah (1) *register* dan *login*, (2) manajemen proyek (CRUD[9]), (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 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 **??**. 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* (**?**). Serangan *brute-*

[9]Create, Read, Update, Delete



Gambar 6. Alur proses *login* ke sistem CCG.

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 **??**. 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 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 **??** 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 **??**. Untuk kepentingan *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 0.1: 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 **??** didapatkan dari fungsi *makeCCGDeriv* yang terdapat pada Kode **??**. 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*. Adapun hasil *render* di sisi *frontend*-nya dapat dilihat pada Gambar **??**.

Kode 0.2: 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": "<" }
```

```
22     ]
23 ]
```

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.

4. Evaluasi

Bagian ini berisi dua sub-bagian, yaitu Hasil Pengujian dan Analisis Hasil Pengujian. Pengujian dan analisis yang dilakukan selaras dengan tujuan TA sebagaimana dinyatakan dalam Pendahuluan.

4.1 Hasil Pengujian

Pertama, tampilkan hasil pengujian yang paling utama. Kemudian hasil-hasil yang lebih detil ditampilkan setelah hasil yang utama. Mengingat tinggi atau rendah, baik atau jeleknya hasil pengujian bersifat relatif, maka sangat dianjurkan ada pembanding (baseline) yang membandingkan dengan algoritma atau pendekatan yang dipilih untuk TA. Pembanding dijalankan pada lingkungan (termasuk data set) yang sama.

Pilih tabel atau jenis diagram yang sesuai untuk menampilkan hasil pengujian.

4.2 Analisis Hasil Pengujian

Analisis merupakan salah satu bagian yang penting untuk TA. Pada TA S1 tidak dituntut untuk mendapatkan hasil performasi yang lebih bagus dibandingkan dengan baseline yang populer, yang dituntut adalah membuat analisis yang lengkap. Menganalisis pengaruh kondisi-kondisi yang berbeda (seperti parameter, jenis data, threshold, dan sub-sistem) yang digunakan.

Cara sitasi adalah sebagai berikut: (**?**) untuk buku, (**?**) untuk *paper*, dan (**?**) untuk website.

5. Kesimpulan

Bagian Kesimpulan memuat kesimpulan dan Saran (*Future Work*), bisa dituliskan dalam poin-poin ataupun paragraf-paragraf. Semua poin kesimpulan diambil dari hasil pengujian dan analisis hasil pengujian sehingga tidak ada kesimpulan dari teori ataupun nalar semata. Sebagaimana sudah disebutkan pada bagian sebelumnya, pengujian dan analisis harus sesuai dengan tujuan TA. Jadi kesimpulan-kesimpulan yang dituliskan selaras dengan seluruh tujuan TA.

Lampiran

Lampiran dapat berupa detil data dan contoh lebih lengkapnya, data-data pendukung, detail hasil pengujian, analisis hasil pengujian, detail hasil survey, surat pernyataan dari tempat studi kasus, screenshot tampilan sistem, hasil kuesioner dan lain-lain.