Terjemahan dari 10.22146/jnteti.v13i2.9184

Entity dan *Relation Linking* untuk *Knowledge Graph Question Answering* Menggunakan Pencarian Berjenjang

Adila Alfa Krisnadhi[1], Mohammad Yani[2], Indra Budi[3]

[1] Program Studi Ilmu Komputer, Fakultas Ilmu Komputer, Universitas Indonesia, Depok, Jawa Barat 16424, Indonesia
[2] Program Studi Rekayasa Perangkat Lunak, Jurusan Teknik Informatika, Politeknik Negeri Indramayu, Indramayu, Jawa Barat 45252, Indonesia
[3] Program Studi Sistem Informasi, Fakultas Ilmu Komputer, Universitas Indonesia, Depok, Jawa Barat 16424, Indonesia

[Diserahkan: 10 Agustus 2023, Direvisi: 12 Oktober 2023, Diterima: 18 Maret 2024]
Penulis Korespondensi: Mohammad Yani (email: mohammad.yani@polindra.ac.id)

INTISARI — Sistem *knowledge graph question answering* (KGQA) berperan penting dalam penarikan data dari *knowledge graph* (KG). Dengan sistem ini, pengguna biasa dapat mengakses data dari KG tanpa perlu membuat kueri SPARQL formal. Sistem KGQA mendapatkan *natural language question* (NLQ) dan menerjemahkannya ke dalam kueri SPARQL menggunakan tiga tugas utama, yaitu deteksi entitas dan relasi, *entity linking* dan *relation linking*, serta konstruksi kueri. Namun, penerjemahan ini tidaklah mudah karena adanya kesenjangan leksikal dan ambiguitas entitas yang dapat terjadi selama proses *entity linking* dan *relation linking*. Untuk mengatasi kesenjangan leksikal tersebut, penelitian ini mengusulkan sebuah pendekatan berdasarkan klasifikasi multikelas NLQ yang kemunculan entitasnya dideteksi menjadi kategori-kategori berdasarkan relasi KG. Selanjutnya, untuk mengatasi ambiguitas entitas, penelitian ini mengusulkan prosedur pencarian tiga tahap untuk menemukan entitas KG yang sesuai dan dapat dikaitkan dengan entitas NLQ berdasarkan korespondensi antara NLQ dan relasi KG tertentu. Tiga tahap pencarian tersebut terdiri atas pencarian berbasis teks, pencarian berbasis vektor, dan pasangan entitas dan relasi. Pendekatan yang diusulkan dievaluasi pada *dataset* SimpleQuestions dan LC-QuAD 2.0. Percobaan menunjukkan bahwa pendekatan yang diusulkan mengungguli *baseline* yang sudah ada. Pada tugas *relation linking*, pendekatan yang diusulkan mencapai *recall* sebesar 89,87% untuk SimpleQuestions dan 74,83% untuk LC-QuAD 2.0. Pada tugas *entity linking*, pendekatan ini juga mencapai *recall* sebesar 91,74% untuk SimpleQuestions dan 61,96% untuk LC-QuAD 2.0.

KATA KUNCI — *Entity Linking*, *Relation Linking*, Sistem KGQA, *Knowledge Graph*.

I. PENDAHULUAN

Penelitian tentang sistem *question answering* (QA) masih menarik dan masif bagi komunitas bahasa alami. Terdapat tiga jenis sistem QA, yaitu *information retrieval question answering* (IRQA), *knowledge graph question answering* (KGQA), dan *hybrid question answering* (HQA) [1]. Perbedaan antara IRQA dan KGQA terletak pada data yang digunakan; IRQA menggunakan teks sebagai datanya, sedangkan KGQA menggunakan *knowledge graph* (KG). Sementara itu, HQA menggunakan kombinasi KG dan teks secara bersamaan.

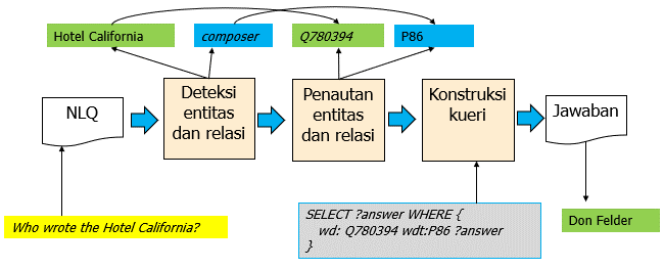
Penelitian KGQA masih menjadi sebuah tantangan tersendiri dalam kurun waktu sepuluh tahun terakhir. KG adalah kumpulan pernyataan fakta yang diberikan dalam bentuk *triple*, yang masing-masing menggambarkan entitas yang saling terkait oleh sebuah relasi. Dalam hal ini, sebuah entitas dalam KG dapat merepresentasikan sebuah objek, peristiwa, atau konsep di dunia nyata. Seseorang biasanya harus menulis kueri formal dalam bahasa formal seperti SPARQL, yang mungkin dirasa sulit bagi pengguna awam, untuk memperoleh data yang diekspresikan oleh pernyataan-pernyataan tersebut. Tanpa harus menggunakan kueri SPARQL, sistem KGQA membantu pengguna awam dengan mengizinkan pengambilan data dari KG menggunakan *natural language question* (NLQ) sebagai masukan. Tugas merumuskan kueri SPARQL yang mengekspresikan masukan NLQ dilakukan oleh sistem KGQA secara otomatis.

Sistem KGQA yang ada saat ini menggunakan konstruktor kueri untuk menarik data dari KG. Konstruktor kueri membutuhkan entitas dan relasi yang tepat serta posisi entitas dan relasi (didapatkan dari *entity linking* dan *relation linking*) untuk memperoleh jawaban yang benar. Namun, model entitas dan relasi yang ada saat ini hanya memberikan set entitas dan relasi dan tidak memberikan posisi entitas dan relasi dalam *triple*. Oleh karena itu, konstruktor kueri harus mencoba semua kemungkinan posisi entitas dan relasi. Sebuah model telah digunakan untuk mengatasi masalah ini [2]. Penelitian ini berfokus pada tugas-tugas *entity linking* dan *relation linking* dengan memanfaatkan hasil deteksi entitas yang diusulkan [2].

II. *KNOWLEDGE GRAPH QUESTION ANSWERING*

Secara teknis, sistem KGQA menerjemahkan NLQ menjadi kueri SPARQL yang kemudian dapat digunakan untuk mengambil data melalui tiga tugas utama, yaitu deteksi entitas dan relasi, *entity linking* dan *relation linking*, serta konstruksi kueri. Penggunaan SPARQL untuk mengambil data dalam KG tidak mudah dilakukan karena sistem harus menjalankan beberapa tugas. Tugas pertama adalah mendeteksi entitas dan relasi pada pertanyaan; kemudian, menghubungkan entitas dan relasi tersebut ke entitas dan relasi pada KG. Tugas terakhir adalah membuat kueri yang benar dengan menggunakan entitas dan relasi yang tepat untuk memperoleh jawaban.

Gambar 1 mengilustrasikan cara kerja sistem KGQA *end-to-end*. Dari pertanyaan (q) “Who wrote the Hotel California?” deteksi entitas dan relasi kemudian mengekstrak entitas dan



Gambar 1. Contoh cara menjalankan sistem KGQA.

relasi yang disebutkan dalam pertanyaan. Pada contoh ini, sistem memperoleh "Hotel California" dan "composer" sebagai entitas dan relasi. Tugas *entity linking* dan *relation linking* adalah menghubungkan entitas dan relasi yang diperoleh dari tugas sebelumnya. Tugas ini menghasilkan "Q780394" dan "P86" sebagai *ID* entitas dan *ID* relasi. Konstruksi kueri membangun kueri formal untuk mengambil data dari KG menggunakan daftar entitas dan relasi yang diperoleh dari tugas *entity linking* dan *relation linking* [3]. Kueri yang dibangun pada tugas tersebut adalah:

```
SELECT ?answer WHERE {
   wd:Q780394 wdt:P86 ?answer .
}
```

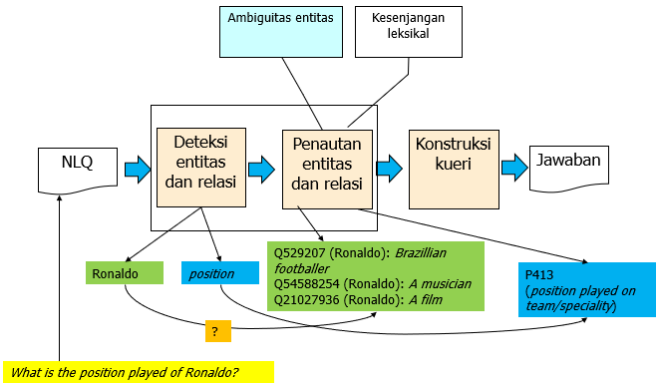
Kueri tersebut ditulis dalam sintaks SPARQL. SPARQL merupakan standar bahasa yang digunakan untuk mengakses data dalam bentuk *resource description framework* (RDF) yang tersimpan dalam sebuah KG [4]. RDF adalah sebuah kerangka kerja yang digunakan untuk mengekspresikan informasi sumber daya, seperti manusia dan dokumen [5].

Dari ketiga tugas tersebut, tugas terakhir akan menjadi lebih mudah jika entitas dan relasi untuk NLQ yang diberikan telah diperoleh. Sementara itu, dua tugas yang pertama lebih sulit. Penelitian sebelumnya telah meneliti cara mengatasi permasalahan tugas deteksi entitas [2]. Tugas pertama (deteksi entitas dan relasi) mengatasi permasalahan pendefinisian posisi entitas yang terdeteksi dalam *triple* untuk suatu NLQ [2]. Model mengekstrak entitas yang disebutkan dalam pertanyaan menggunakan pendekatan pola berbasis posisi. Menggunakan pendekatan ini, model menghasilkan satu set entitas dan posisi dalam *triple*. Hasil dari model ini dapat membantu tugas *entity linking* dan *relation linking* untuk menemukan entitas dan relasi yang tepat dalam KG.

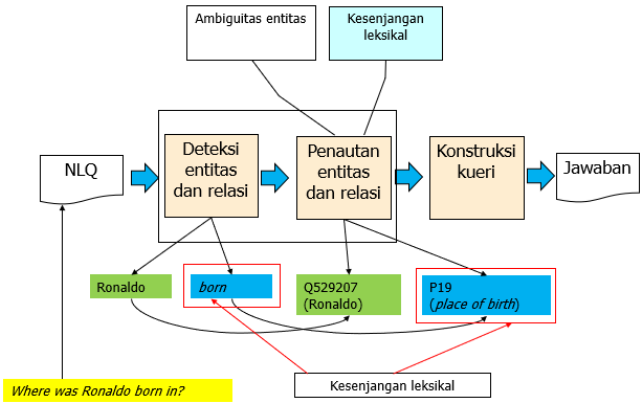
Penelitian ini berfokus pada tugas *entity linking* dan *relation linking*. Sebagai tambahan, berdasarkan pendekatan yang diusulkan ini, tugas pendeteksian relasi dapat sepenuhnya dilewati karena tugas *entity linking* mampu mengaitkan relasi KG ke NLQ ketika semua entitas telah terdeteksi.

Dua tantangan utama yang perlu diatasi sehubungan dengan *entity linking* dan *relation linking* adalah ambiguitas entitas dan kesenjangan leksikal [6]. Ambiguitas entitas mengacu pada entitas dalam NLQ yang dapat dikaitkan dengan entitas KG dengan nama yang sama, tetapi memiliki arti yang berbeda (dikenal sebagai polisemi) [7], [8]. Sementara itu, kesenjangan leksikal mengacu pada kasus ketika bentuk permukaan entitas atau relasi dalam pertanyaan sama sekali berbeda dengan bentuk permukaan entitas dan relasi dalam KG. Gambar 2 dan Gambar 3 mengilustrasikan tantangan-tantangan ini dengan contoh-contoh yang lebih konkret.

Gambar 2 menunjukkan bahwa terdapat tiga entitas bernama "Ronaldo." Setiap entitas memiliki label yang sama tetapi memiliki arti yang berbeda. Entitas dengan *ID* Q529207,



Gambar 2. Ilustrasi ambiguitas entitas.



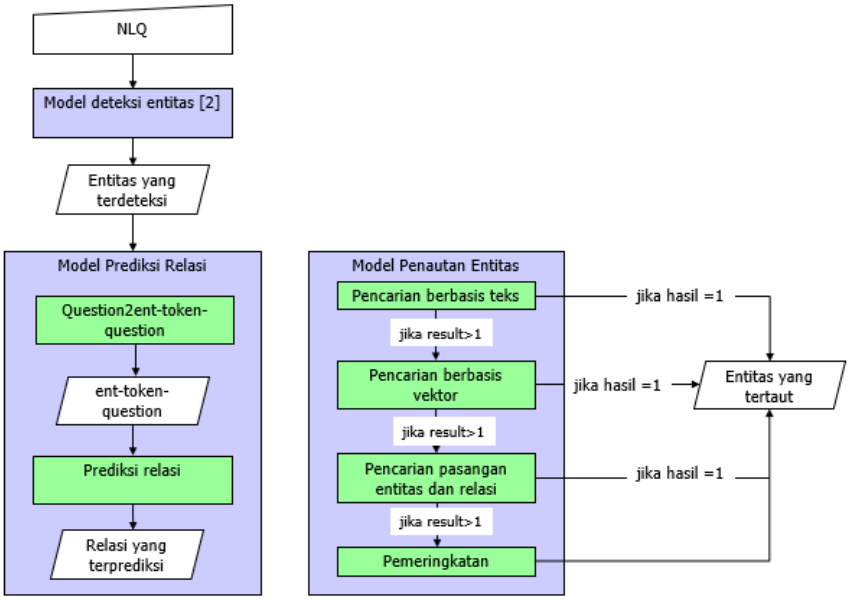
Gambar 3. Ilustrasi kesenjangan leksikal.

Q54588254, Q21027936 masing-masing merepresentasikan "Ronaldo" sebagai seorang pemain sepak bola Brazil, seorang musisi, dan sebuah judul film. Namun, entitas yang disebutkan dalam pertanyaan adalah "Ronaldo" seorang pemain sepak bola Brazil (Q529207). Meskipun masalah ini mudah diatasi oleh manusia, tetapi tidak demikian halnya dengan mesin. Masalah ini mencapai 60% dari semua permasalahan pada sistem KGQA [6].

Gambar 3 mengilustrasikan permasalahan kesenjangan leksikal dalam tugas *relation linking*. Pada contoh tersebut, KG mengekspresikan relasi "born" dengan leksikal yang berbeda, yaitu "place of birth." Permasalahan ini menyebabkan kesalahan kait dalam tugas *relation linking*. Masalah kesenjangan leksikal merupakan masalah yang paling banyak terjadi pada sistem KGQA, yaitu sebesar 65,7% [6].

Setidaknya terdapat dua pendekatan untuk mengatasi permasalahan ini. Pada pendekatan pertama, *entity linking* dan *relation linking* dijalankan secara independen [9]–[11]. Apa pun hasil yang diperoleh dari kedua tugas tersebut digunakan untuk membangun kueri KG yang sesuai. Kekurangan dari pendekatan ini adalah independensi antara *entity linking* dan *relation linking*. Independensi ini menyebabkan *entity linking* tidak dapat memanfaatkan informasi relasi dari *relation linking* yang mungkin berguna. Oleh karena itu, ketidakcocokan dapat berpotensi terjadi karena ambiguitas dalam nama/sebutan entitas.

Pada pendekatan kedua, entitas dan relasi dipasangkan. Tujuannya adalah agar keduanya dapat secara bersamaan menemukan entitas dan relasi KG yang dapat dicocokkan dengan entitas dan relasi yang disebutkan dalam masukan NLQ. Penelitian-penelitian sebelumnya yang menggunakan pendekatan ini tidak menggunakan informasi semantik tambahan yang dapat diperoleh dari deskripsi entitas dan nama alias yang tersedia dalam KG [12]–[14]. Hal ini merupakan



Gambar 4. Arsitektur umum dari pendekatan yang diusulkan.

sebuah kelemahan karena, pada prinsipnya, proses harus mempertimbangkan secara mendalam segala kemungkinan pasangan entitas-relasi dalam KG.

Sebagai alternatif dari pendekatan-pendekatan di atas, *relation linking* diusulkan untuk dilakukan terlebih dahulu sebelum *entity linking*. Lebih tepatnya, *relation linking* dimodelkan sebagai klasifikasi multikelas yang targetnya berasal dari relasi KG dan masukannya adalah *masked NLQ*, yaitu masukan NLQ yang penyebutan semua entitasnya disamarkan menggunakan token [ENT] khusus. Pendekatan ini memanfaatkan fakta bahwa penyebutan entitas telah terdeteksi pada langkah deteksi entitas sebelumnya (solusi yang telah dijelaskan dalam penelitian sebelumnya [2]). Selain itu, setelah masukan NLQ dicocokkan dengan relasi yang sesuai, *entity linking* dilakukan melalui prosedur pencarian tiga langkah: pencarian berbasis teks, pencarian berbasis vektor, dan pasangan entitas dan relasi. Kontribusi utama dari pendekatan ini adalah kemampuannya memisahkan entitas dalam KG dengan memasangkan entitas dan relasi dengan baik dan dapat mengidentifikasi relasi yang tepat dalam KG meskipun diekspresikan dalam bentuk permukaan yang berbeda dari relasi dalam NLQ.

Makalah ini disusun sebagai berikut. Bagian Metodologi menyajikan pendekatan yang diusulkan untuk mengatasi masalah *entity linking* dan *relation linking*, seperti yang telah dijelaskan di atas. Bagian Hasil dan Diskusi menyajikan hasil penelitian yang dibandingkan dengan pendekatan lain. Bagian akhir menyimpulkan penelitian ini dan memperkenalkan penelitian selanjutnya.

III. METODOLOGI

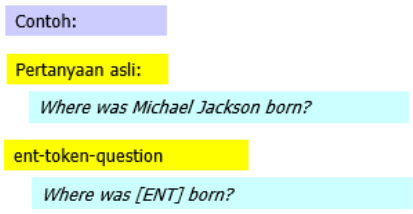
Tiga metode yang ada digunakan untuk mengimplementasikan tugas *entity linking* dan *relation linking*, yaitu pendekatan berurut, paralel, dan gabungan entitas-relasi. Pendekatan berurut bekerja dengan mengaitkan entitas secara berurutan, kemudian mengaitkan relasi. Pendekatan paralel melakukan kedua tugas tersebut secara bersamaan. Pendekatan gabungan relasi-entitas memasangkan relasi-entitas untuk menghubungkannya ke dalam sebuah KG [10]–[16].

Pendekatan yang diusulkan terdiri atas dua subtugas, yaitu tugas *relation linking* dan *entity linking*. Tugas-tugas ini bertujuan untuk menghubungkan entitas dan relasi yang diekstrak dari pertanyaan-pertanyaan dengan entitas dan relasi dalam KG. Tugas-tugas tersebut dilakukan secara berurutan. Pendekatan yang diusulkan ini melakukan *relation linking* terlebih dahulu, kemudian melakukan *entity linking*. Hal ini berbeda dengan pendekatan berurutan yang ada saat ini; pendekatan ini melakukan *entity linking* sebelum *relation linking*. Pendekatan ini dipilih karena masalah ambiguitas entitas yang terjadi pada pendekatan berurutan yang ada saat ini. Pendekatan tersebut tidak dapat memetakan relasi sebagai pasangan dari entitas. Selain itu, masalah kesenjangan leksikal juga terjadi pada pendekatan berurutan yang ada, yaitu sebesar 65,7% [6].

Pendekatan yang diusulkan ini menggunakan Elasticsearch sebagai mesin pencari lokal. Dua jenis mesin pencari lokal diusulkan, yaitu mesin pencari berbasis teks dan mesin pencari berbasis vektor. Mesin pencari ini mengkodekan *ID* entitas dan label Wikidata dari timbunan Wikidata ke dalam bentuk indeks di Elasticsearch [17]. Perbedaannya terletak pada bentuk data yang disimpan dalam Elasticsearch; mesin pencari lokal berbasis vektor mengkodekan data dalam bentuk vektor, bukan teks.

Gambar 4 menunjukkan arsitektur umum pendekatan yang diusulkan untuk *entity linking* dan *relation linking* untuk sistem KGQA. Sebagai contoh, sebuah NLQ diberikan kepada sistem ini. Pertama, entitas-entitas yang disebutkan dalam NLQ diekstraksi menggunakan model pendeteksian entitas yang diusulkan sebelumnya [2]. Model ini juga memberikan informasi tentang posisi entitas dalam *triple*. Informasi posisi yang diberikan berisi posisi entitas dalam kueri dan posisi entitas dalam *triple*. Posisi entitas dalam kueri menunjukkan posisi entitas pada *triple*, yakni di *triple* pertama, kedua, atau ketiga. Sementara itu, posisi entitas di dalam *triple* menunjukkan posisi entitas, yaitu di bagian kepala atau ekor.

Sebagaimana ditunjukkan pada Gambar 4, nama entitas yang diekstrak dari pertanyaan digunakan untuk membuat ent-token-question, yang mana token/kata yang sesuai dengan nama entitas diubah menjadi token [ENT]. Model *bidirectional encoder representations from transformer* (BERT) digunakan



Gambar 5. Ilustrasi konversi pertanyaan menjadi ent-token-question.

untuk mengimplementasikan tugas klasifikasi dalam memprediksi suatu relasi. Model deteksi entitas dalam [2] digunakan untuk menghubungkan entitas yang diekstrak dari sebuah pertanyaan. Terdapat tiga pendekatan dalam menghubungkan entitas, yaitu pencarian berbasis teks, pencarian berbasis vektor, dan pencarian pasangan entitas-relasi. Proses pencarian dilakukan secara berurutan. Proses selanjutnya dijalankan ketika jumlah kandidat entitas lebih dari satu. Pada subtugas terakhir, *cosine similarity* digunakan untuk menilai kemiripan semua kandidat entitas.

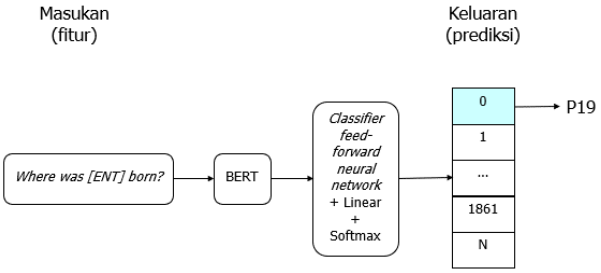
A. RELATION LINKING

Tujuan dari tugas ini adalah untuk menghubungkan relasi yang digunakan dalam pertanyaan dengan relasi dalam KG. Secara konvensional, tugas ini menerima masukan dari tugas pendeteksian relasi berupa kata/token yang diidentifikasi sebagai relasi dalam pertanyaan.

Klasifikasi multikelas digunakan untuk memodelkan tugas *relation linking*. Masukan dari masalah ini adalah *masked NLQ* yang disebut ent-token-question. Sebuah ent-token-question mirip dengan NLQ yang memiliki masukan biasa, hanya saja penyebutan entitasnya disamarkan oleh token [ENT] khusus. Target klasifikasi terdiri atas relasi KG, pasangan relasi tersebut, dan set yang berisi tiga relasi tersebut. Pada praktiknya, tidak semua kemungkinan kombinasi pasangan relasi/*triple* dipertimbangkan; hanya pasangan relasi atau *triple* yang terkait dengan pertanyaan yang sama sesuai dengan *dataset* yang digunakan untuk membangun model klasifikasi saja yang dipertimbangkan. Pengubahan pertanyaan asli menjadi ent-token-question dilakukan pada tahap prapemrosesan.

ent-token-question dipilih karena kemampuannya untuk mempelajari banyak ekspresi yang digunakan dalam menyatakan relasi pada *dataset* SimpleQuestions dan LC-QuAD 2.0. Selain itu, tujuan utama menyembunyikan penyebutan entitas dalam NLQ adalah untuk menekankan ekspresi relasi tanpa bergantung pada sebutan/nama entitas dalam pertanyaan. Gambar 5 menunjukkan contoh ent-token-question "Where was [ENT] born?" yang mewakili relasi "place of birth" (P19). Sebelum pelatihan, *dataset* yang berisi pertanyaan asli dikonversi ke dalam bentuk yang berisi relasi ent-token-question dan *ID* di kolom pertama dan kedua. Kolom pertama digunakan sebagai fitur masukan model, sedangkan kolom kedua digunakan sebagai label.

Selain itu, penggunaan kata tanya "what," "when," dan "where" juga dianggap mengindikasikan hubungan yang disebutkan dalam pertanyaan. Masih dalam contoh yang sama, pertanyaan "Where was [ENT] born?" mungkin menghasilkan "place of birth" (P19) dan "date of birth" (P569). Pendekatan yang diusulkan ini dapat mengidentifikasi "place of birth" (P19) sebagai relasi yang benar yang disebutkan dalam pertanyaan dengan mempertimbangkan kata tanya yang digunakan dalam pertanyaan.



Gambar 6. Prediksi relasi menggunakan tugas multi-kelas klasifikasi.

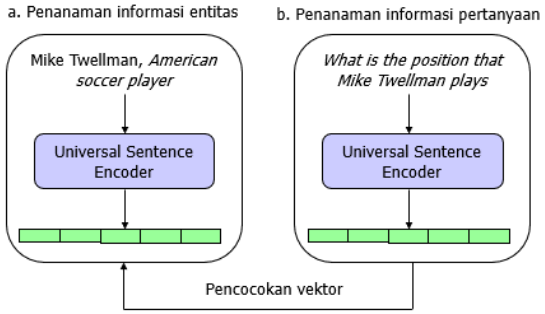
BERT yang telah disesuaikan [18] digunakan untuk memprediksi relasi dari pertanyaan yang diberikan. Gambar 6 menunjukkan masukan dan keluaran model dalam memprediksi hubungan *ID* dari pertanyaan yang diberikan. Model *transformer* sederhana digunakan dalam pelatihan karena adanya keterbatasan sumber daya komputasi [19]. *transformer* sederhana menyederhanakan *face library* asli [20] tanpa menghilangkan substansinya.

B. ENTITY LINKING

Tugas ini terdiri atas tiga subtugas, yaitu pencarian berbasis teks, pencarian berbasis vektor, dan tugas-tugas pemasangan entitas dan relasi. Pencarian berbasis teks menemukan kandidat entitas dengan mencocokkan nama label entitas dalam pertanyaan dan nama label entitas dalam KG. Sebagai contoh, semua entitas yang bernama "Joe Biden" akan dikembalikan ketika entitas bernama "Joe Biden" diminta.

Pencarian berbasis vektor digunakan jika pencarian berbasis teks tidak menghasilkan nilai kembali (*return value*) atau menghasilkan lebih dari satu entitas. Pendekatan ini mencocokkan nilai vektor dari entitas yang disebutkan dalam pertanyaan dengan informasi entitas dalam KG. Informasi entitas yang dikodekan dalam KG adalah nama label, deskripsi, dan nama alias. Sementara itu, informasi yang dikodekan dalam pertanyaan adalah semua frasa atau kata yang disebutkan dalam pertanyaan. Masih dalam contoh yang sama untuk pencarian "Joe Biden," pencarian berbasis vektor akan mengembalikan semua entitas yang berhubungan dengan penyebutan entitas tersebut dalam pertanyaan. Dengan contoh yang sama, pencarian berbasis teks dan pencarian berbasis vektor mendapatkan dua set entitas yang berbeda, yaitu masing-masing [Q6279, Q65053339] dan [Q6279, Q65053339, ..., Q129756]. Dalam contoh ini, penggunaan pencarian berbasis vektor menghasilkan ukuran set yang lebih besar daripada pencarian berbasis teks. Ukuran himpunan yang besar pada pencarian berbasis vektor memiliki kelebihan dan kekurangan. Kelebihannya adalah himpunan tersebut dapat berisi entitas yang tidak ditemukan dalam himpunan pencarian berbasis teks, sedangkan kekurangannya adalah penggunaan set ini membuat ruang pencarian menjadi lebih luas. *Universal sentence encoder* [21] digunakan untuk memberikan kode informasi. Gambar 7 menunjukkan cara beroperasi pencocokan vektor.

Pasangan entitas dan relasi digunakan jika pendekatan pencarian berbasis vektor menghasilkan lebih dari satu entitas (termasuk entitas yang ambigu). Pendekatan ini bertujuan untuk memeriksa relasi tertentu sebuah entitas dengan entitas lain. Jika nilai kembali adalah True, entitas tersebut diasumsikan sebagai entitas yang tepat, demikian juga sebaliknya. Namun, jika beberapa entitas memiliki relasi tertentu, kandidat entitas teratas yang akan dipilih. Sebagai contoh, pertanyaan "What is the position that Mike Twellman plays" menghasilkan satu set kandidat entitas [Q6849115,



Gambar 7. Pencarian berbasis vektor.

Q5921964, ..., Q5059480]. Jadi, setiap kandidat entitas diperiksa untuk memastikan ada tidaknya relasi yang diperoleh dari tugas *relation linking*. Jika entitas tersebut memiliki relasi yang diperoleh dari tugas *relation linking*, entitas tersebut diasumsikan sebagai entitas yang benar yang disebutkan di dalam pertanyaan. Jika tidak, entitas tersebut dieliminasi.

Dalam subtugas *relation linking*, Question2ent-token-question mengubah pertanyaan menjadi format ent-token-question. Sebagai contoh, pertanyaan "What is the position that Mike Twellman plays" diubah menjadi "What is the position that [ENT] plays." Token "Mike Twellman" diubah menjadi "[ENT]" karena token tersebut adalah sebuah entitas. Tugas konstruksi data mengumpulkan semua ent-token-question yang berasal dari tugas Question2ent-token-question. Data yang dikonstruksi terdiri atas dua bagian, yaitu ent-token-question dan label. Mesin kemudian melatih data tersebut menggunakan klasifikasi multikelas dengan model BERT.

Tiga subtugas untuk tugas *entity linking* diusulkan. Pencarian berbasis teks mencari entitas yang tepat dengan mencocokkan *string* menggunakan *exact matching* dan *substring matching*. Pencarian berbasis vektor digunakan ketika hasil pencarian berbasis teks tidak menemukan *string* yang cocok. Tugas ini mengukur kemiripan dua *string* yang disematkan dalam ruang vektor. Subtugas terakhir adalah tugas pencocokan relasi-entitas. Tugas ini mencari entitas yang memiliki pasangan relasi yang diperoleh dari tugas *relation linking*.

IV. HASIL DAN DISKUSI

Bagian ini menyajikan pengaturan penelitian, metode evaluasi, serta hasil dan diskusi dari penelitian ini. Hasil penelitian dibandingkan dengan *baseline* untuk *entity linking* dan *relation linking* dengan menggunakan *dataset* dan KG yang sama.

Pada bagian evaluasi, SimpleQuestions dan LC-QuAD 2.0 digunakan karena sistem KGQA yang ada saat ini masih memiliki kekurangan pada *dataset*, sebagaimana telah dijelaskan pada bagian Pendahuluan. Selain itu, penelitian ini hanya membahas masalah ambiguitas entitas dan kesenjangan leksikal dalam sistem KGQA. Makalah ini hanya membahas masalah pertanyaan yang memuat maksimal tiga *triple*.

A. SIMPLEQUESTIONS DAN LC-QUAD 2.0

SimpleQuestions berisi satu set pertanyaan sederhana. Pertanyaan sederhana adalah pertanyaan yang hanya terdiri atas satu *triple*. *Triple* terdiri atas subjek, predikat, dan objek [5]. SimpleQuestions memiliki dua versi, yaitu versi *freebase* [22] dan versi Wikidata [23]. Wikidata SimpleQuestions berisi 27.924 pertanyaan.

LC-QuAD 2.0 adalah *dataset* yang berisi pertanyaan kompleks. Pertanyaan kompleks adalah pertanyaan yang mengandung lebih dari satu *triple*. *Dataset* ini terdiri atas sepuluh jenis pertanyaan yang berbeda, termasuk pertanyaan sederhana dan kompleks, seperti pertanyaan multifaktor. *Dataset* ini berisi 30 juta pertanyaan dan kueri kebenaran dasar. Kebenaran dasar berisi kueri SPARQL yang merepresentasikan jawaban dari pertanyaan tersebut [24].

B. PENGATURAN PENELITIAN

Bagian ini menjelaskan pengaturan penelitian yang digunakan untuk mengevaluasi pendekatan yang diusulkan, yaitu, alat pengekstraksi entitas, spesifikasi mesin, *dataset*, KG, dan konfigurasi data serta parameter yang digunakan.

Alat pengekstraksi entitas yang diusulkan, yang disebut metode *position-based pattern* (PBP) [2], digunakan untuk mengekstrak nama entitas dari pertanyaan. Pola berbasis posisi merepresentasikan sebuah pasangan (χ, ℓ), dengan χ menunjukkan posisi kepala atau ekor dan ℓ adalah posisi berbasis token. Metode ini mengekstrak entitas yang disebutkan dalam pertanyaan dan memberikan posisi entitas dalam *triple*. *Triple* adalah bentuk yang digunakan oleh KG untuk merepresentasikan sebuah fakta.

Keluaran dari PBP adalah kumpulan PBP *Cq*. Digunakan pertanyaan *q* "Di mana John Morris Russel lahir?" untuk mengilustrasikan set pola berbasis posisi. Token "John Morris Russell" diposisikan dalam indeks [2, 3, 4] (dimulai dengan 0) dalam *q*. Dengan demikian, set PBP dari *q* adalah sebagai berikut.

$$Cq = \{0: (head, [2,3,4])\}$$

dengan bilangan bulat 0 menunjukkan bahwa token "John Morris Russell" berada di *triple* pertama. Label *head* menunjukkan bahwa "John Morris Russell" memiliki pola kepala. Pola kepala adalah pola yang entitasnya berada pada posisi subjek. Untuk tugas klasifikasi, *Cq* dikodekan menjadi *0:head:ent:2_3_4*. *Cq* yang dikodekan digunakan sebagai label *q* dalam tugas klasifikasi multikelas.

Prediksi pola berbasis posisi menggunakan pengklasifikasi multikelas yang beroperasi di bawah model *transformer*. Masukan model tersebut adalah sebuah pertanyaan, sedangkan keluarannya berupa prediksi PBP dari pertanyaan yang diberikan. Keluaran PBP digunakan sebagai masukan pendekatan yang diusulkan untuk menghubungkan entitas dan relasi yang disebutkan dalam pertanyaan serta entitas dan relasi dalam KG. Prapemrosesan dilakukan sebelum menggunakan PBP untuk tugas *entity linking*. Prapemrosesan mengubah set bilangan bulat dari PBP menjadi kata/token yang sesuai dengan posisi kata/token di dalam pertanyaan. Set kata/token yang diperoleh kemudian diubah ke dalam bentuk ent-token-question.

Penelitian ini menggunakan GPU NVIDIA GeForce GTX dengan memori 24 GB untuk pelatihan data. Sementara itu, CPU 3.0 GHz Intel(R) i7-5960X dengan memori 128 GB digunakan untuk prapemrosesan data. Penelitian ini menggunakan SimpleQuestions pada Wikidata [23] dan tidak menggunakan SimpleQuestions asli [22] karena SimpleQuestions asli tidak menggunakan Wikidata sebagai KG. Penelitian ini juga menggunakan LC-QuAD 2.0 [24] pada Wikidata untuk pertanyaan-pertanyaan yang kompleks. Wikidata berisi jutaan entitas yang dapat disunting oleh siapa saja. Data dalam Wikidata disimpan dalam format RDF [25].

Penelitian ini menggunakan pengaturan bawaan untuk membagi *dataset* menjadi data pelatihan, validasi, dan

TABEL I
DISTRIBUSI DATA SIMPLEQUESTIONS DAN LC-QUAD 2.0

Dataset	Data Latih	Data Validasi	Data Uji	Total
SimpleQuestions	19.481	2.821	5.622	27.924
LC-QuAD 2.0	22.132	3.306	6.383	31.821

pengujian. Komposisi pemisahan ini digunakan karena dapat dengan adil dibandingkan menggunakan Falcon 2.0. Tabel I menyajikan distribusi *dataset* SimpleQuestions dan LC-QuAD 2.0. *Dataset* LC-QuAD 2.0 tidak menetapkan set validasi terpisah, sehingga diperoleh data validasi sebesar 13% dari keseluruhan data pelatihan.

Penelitian ini menggunakan Wikidata sebagai KG. Fakta yang digunakan dalam penelitian ini adalah fakta dari timbunan Wikidata. Timbunan tersebut diunduh dari https://dumps.wikimedia.org/ pada tahun 2019. Timbunan berisi sekitar 49 juta entitas dan 6 ribu relasi.

C. METRIK EVALUASI

Model yang diusulkan menghasilkan entitas dan relasi serta posisinya dalam *triple*. Namun, model entitas dan relasi yang ada, seperti Falcon 2.0, tidak memberikan posisi entitas dan relasi dalam *triple*. Oleh karena itu, untuk membandingkan secara adil pendekatan yang diusulkan dengan Falcon 2.0, informasi posisi entitas dan relasi dalam *triple* tidak digunakan pada evaluasi ini.

Evaluasi tersebut menggunakan pengukuran berbasis *recall* dari entitas dan relasi yang diprediksi. Berikut adalah beberapa alasan penggunaan pengukuran berbasis *recall*.
Intuisi:

a. *entity linking* dan *relation linking* yang diprediksi dapat menghasilkan banyak entitas atau relasi;
b. kebenaran dasar juga mengandung beberapa entitas atau relasi;
c. membangun kueri *entity linking* dan *relation linking* yang diprediksi agar sesuai dengan kueri yang tepat tidaklah mudah;
d. oleh karena itu, akan lebih baik jika hasil *entity linking* dan *relation linking* mencakup sebanyak mungkin entitas atau relasi kebenaran dasar;
e. oleh karena itu, pengukuran berdasarkan *recall* lebih tepat digunakan.

Berikut adalah formula yang digunakan. q adalah sebuah pertanyaan dan Y_q^* adalah set entitas atau relasi kebenaran dasar. Jika model *entity linking* atau *relation linking* menghasilkan Y_q sebagai set entitas atau relasi yang diprediksi, pengukuran berbasis *recall* untuk q adalah:

$$score(q) = \frac{|Y_q^* \cap Y_q|}{|Y_q^*|}. \quad (1)$$

Kinerja model kemudian dihitung menggunakan *average recall* (AR) pada set uji $\mathcal{D}$:

$$AR(\mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{q \in \mathcal{D}} score(q). \quad (2)$$

D. MODEL BASELINE

Penelitian ini menggunakan Falcon 2.0 sebagai dasar karena kinerjanya mengungguli *baseline* lain, sebagaimana dijelaskan di bagian Pendahuluan. Selain itu, *baseline* tidak menyediakan kode sumber atau API apa pun untuk mereplikasi hasilnya. Sebaliknya, Falcon 2.0 menyediakan API untuk mereplikasi hasil *dataset* tertentu. Oleh karena itu, dalam percobaan ini, Falcon 2.0 dipilih sebagai *baseline*.

TABEL II
PERBANDINGAN *RECALL* TUGAS *RELATION LINKING* ANTARA PENDEKATAN YANG DIUSULKAN DAN FALCON 2.0

Dataset	Falcon 2.0 (%)	Pendekatan yang Diusulkan (%)
Diuji pada SimpleQuestions	41,33	91,74
Diuji pada LC-QuAD 2.0	27,75	61,96

Pertanyaan (q): *When did Joko Widodo marry Iriana?*

Pola Triple (Tq) atau *Ground-truth*:

```
SELECT ?value1 ?value2 WHERE
{
wd:Q33118231 p:P26 ?s .      ←— r1: P26
?s ps:P26 wd:Q17410605 .     ←— r2: P26
?s pq:P580 ?value1 .         ←— r3: P580
?s pq:P2842 ?value2          ←— r4: P2842
}
```

Gambar 8. Contoh pertanyaan dengan relasi tersembunyi dan tanpa indikator kata kunci pada LC-QuAD 2.0.

E. HASIL RELATION LINKING

Pengukuran berbasis *recall* pada (1) digunakan untuk mengevaluasi metode *relation linking* yang diusulkan. Misalnya, jika kebenaran dasar dan prediksi berisi {'P19', 'P20'}, skor perolehan *recall*-nya adalah 100%. Namun, jika prediksinya {'P19'}, skor *recall*-nya hanya 50%.

Evaluasi ini membandingkan pendekatan yang diusulkan (dengan deteksi entitas berbasis PBP) dengan *baseline*, yaitu Falcon 2.0 [14]. Perbandingan antara pendekatan yang diusulkan dengan Falcon 2.0 dapat dilihat pada Tabel II. Dari tabel ini, terlihat bahwa pendekatan yang diusulkan ini mengungguli Falcon 2.0 dalam hal *recall*. Mengubah pertanyaan asli menjadi ent-token-question dapat mewakili ekspresi relasi yang digunakan dalam pertanyaan tanpa bergantung pada nama entitas. Metode ini dapat mengatasi masalah kesenjangan leksikal yang terjadi dalam tugas *relation linking*. Misalnya, pendekatan yang diusulkan ini dapat memprediksi hubungan "place of birth" (P19) dari pertanyaan yang diberikan "What female actor was born in [ENT]."

Pendekatan yang diusulkan juga mengungguli Falcon 2.0 dalam LC-QuAD 2.0. Namun, *recall* yang dihasilkan lebih rendah dibandingkan SimpleQuestions. Hal ini terjadi karena LC-QuAD 2.0 memiliki banyak pertanyaan, tetapi relasinya tidak didefinisikan secara eksplisit dalam pertanyaan (relasinya tersembunyi).

Pendekatan yang diusulkan tidak dapat menjawab pertanyaan dengan relasi yang tersembunyi dan yang tidak mempunyai indikator kata kunci. Gambar 8 memberikan contoh pertanyaan dengan relasi tersembunyi tanpa kata kunci yang merujuk pada relasi tersebut. Pada gambar tersebut, "r1...r4" mewakili relasi yang digunakan dalam pertanyaan. Relasi "spouse" (P26) dan "start time" (P580) tidak disebutkan secara eksplisit dalam pertanyaan, tetapi terdapat kata kunci indikator yang merujuk pada relasi tersebut, yaitu "when" untuk "start time" dan "marry" untuk "marry." Dalam hal ini, pendekatan yang diusulkan dapat mengatasi masalah tersebut. Namun, untuk relasi tersembunyi tanpa indikator kata kunci, seperti "place of marriage" (P2842), sulit diselesaikan dengan menggunakan pendekatan yang diusulkan.

Falcon 2.0 memiliki *recall* yang rendah karena penggunaan *n-gram tiling* untuk mencocokkan bentuk permukaan relasi dalam pertanyaan dan nama label relasi di KG. Oleh karena itu, Falcon 2.0 tidak dapat mengatasi masalah kesenjangan leksikal. Misalnya, Falcon 2.0 tidak dapat mengidentifikasi kata/token "born" dalam "Who was born in the city of San Francisco" sebagai kaitannya dengan "place of birth" di KG. Tabel III

TABEL III
JUMLAH PERTANYAAN YANG DAPAT DIJAWAB OLEH PENDEKATAN YANG DIUSULKAN TETAPI TIDAK DAPAT DIJAWAB OLEH FALCON 2.0

Dataset	Kualitas
SimpleQuestions	2,866
LC-QuAD 2.0	1,948

TABEL IV
CONTOH PERTANYAAN SIMPLEQUESTIONS YANG DAPAT DIJAWAB OLEH PENDEKATAN YANG DIUSULKAN TETAPI TIDAK DAPAT DIJAWAB OLEH FALCON 2.0

Pertanyaan	Kebenaran Dasar	Pendekatan yang Diusulkan	Falcon 2.0
Who was born in the city of San Francisco?	p19	p19	p131
What was the cause of death of Yves Klein?	p509	p509	p20

TABEL V
CONTOH PERTANYAAN LC-QUAD 2.0 YANG DAPAT DIJAWAB OLEH PENDEKATAN YANG DIUSULKAN TETAPI TIDAK DAPAT DIJAWAB OLEH FALCON 2.0

Pertanyaan	Kebenaran Dasar	Pendekatan yang Diusulkan	Falcon 2.0
What are the characters that appear in Nastes?	p1441 AND p674	p1441 AND p674	p953 AND p674
What genre film was the prequel to Zork II?	p156 AND p136	p156 AND p136	p155

sampai Tabel V menunjukkan pertanyaan-pertanyaan pada tugas *relation linking* yang dapat dijawab dengan pendekatan yang diusulkan tetapi tidak dapat dijawab dengan Falcon 2.0.

F. HASIL ENTITY LINKING

Penelitian ini mengevaluasi model pada *dataset* SimpleQuestions dan LC-QuAD 2.0. Sebaran datanya dapat dilihat pada Tabel I. *Exact matching* digunakan untuk mengevaluasi *recall* prediksi dengan mencocokkan entitas dan urutan antara prediksi dan kebenaran dasar. Hasil dikatakan benar jika entitas dan urutan prediksi benar-benar cocok dengan kebenaran dasar. Misalnya, pertanyaan "Where did Roger Marquis die" memiliki kebenaran dasar {0:{'head':'Q7358590'}}. Jadi, model memperoleh prediksi yang benar jika prediksinya juga {0:{'head':'Q7358590'}}. Penelitian ini tidak membahas masalah lain di luar kosakata. Misalnya, jika entitas "Anas Yani" ada di pertanyaan, tetapi tidak ada di KG, maka entitas "Anas Yani" merupakan masalah di luar kosakata. Untuk kasus tersebut, model yang diusulkan mampu menemukan kemiripan yang paling dekat dengan nama tersebut, misalnya "Malik Ibn Anas."

Pada evaluasi ini dilakukan beberapa percobaan dengan mengonfigurasi nilai ambang batas agar *entity linking* mencapai kinerja yang baik. Percobaan menunjukkan bahwa penggunaan ambang batas 0,2 dapat menghasilkan *recall* terbaik. Tabel VI menunjukkan hasil percobaan dalam menentukan nilai ambang batas terbaik untuk tugas *entity linking*.

Bagian ini membandingkan pendekatan yang diusulkan ditambah PBP dengan Falcon 2.0. Tabel VII menunjukkan perbandingan *recall* antara pendekatan yang diusulkan dan Falcon 2.0. Hasil percobaan menunjukkan bahwa pendekatan yang diusulkan mengungguli Falcon 2.0.

TABEL VI
EKSPERIMEN NILAI AMBANG PADA *ENTITY LINKING*

Nilai Ambang	*Recall* (%)
0,1	89,86
0,2	89,86
0,3	89,86
0,4	89,86
0,5	89,86
0,6	89,86
0,7	89,86
0,8	89,79
0,9	88,97

TABEL VII
PERBANDINGAN *RECALL* TUGAS *ENTITY LINKING* ANTARA PENDEKATAN YANG DIUSULKAN DAN FALCON 2.0

Dataset	Falcon 2.0 (%)	Pendekatan yang Diusulkan (%)
Diuji pada SimpleQuestions	55,69	89,87
Diuji pada LC-QuAD 2.0	39,70	74,83

TABEL VIII
JUMLAH PERTANYAAN YANG DAPAT DIJAWAB OLEH PENDEKATAN YANG DIUSULKAN TETAPI TIDAK DAPAT DIJAWAB OLEH FALCON 2.0

Dataset	Kualitas
SimpleQuestions	2,079
LC-QuAD 2.0	1,916

TABEL IX
CONTOH PERTANYAAN SIMPLEQUESTIONS YANG DAPAT DIJAWAB OLEH PENDEKATAN YANG DIUSULKAN TETAPI TIDAK DAPAT DIJAWAB OLEH FALCON 2.0

Pertanyaan	Kebenaran Dasar	Pendekatan yang Diusulkan	Falcon 2.0
Where did Vic Frazier die?	q7924799	q7924799	q36687
What is Alain Sutter position?	q503421	q503421	q65602988

TABEL X
CONTOH PERTANYAAN LC-QUAD 2.0 YANG DAPAT DIJAWAB OLEH PENDEKATAN YANG DIUSULKAN TETAPI TIDAK DAPAT DIJAWAB OLEH FALCON 2.0

Pertanyaan	Pertanyaan	Pendekatan yang Diusulkan	Falcon 2.0
When did Ashton Kutcher divorce Demi Moore?	q164782 AND q43044	q164782 AND q43044	Empty
What is the fashion house of Alexander McQueen?	q207939 AND q3661311	q207939 AND q3661311	q17659819

Secara umum, pada *entity linking*, *recall* dari pendekatan yang diusulkan mengungguli Falcon 2.0 dengan peningkatan sekitar 30%. Falcon 2.0 memiliki skor *recall* rendah karena Falcon 2.0 tidak cukup baik dalam tugas *relation linking*. Oleh karena itu, pemasangan entitas dan relasi yang dilakukan oleh Falcon 2.0 menghasilkan fakta yang salah. Oleh karena itu, Falcon 2.0 tidak dapat menjelaskan dengan baik masalah ambiguitas entitas. Misalnya, pertanyaan "What is Alain Sutter position played in" menampilkan dua entitas bernama "Alain Sutter" di KG, yaitu, "Alain Sutter" (Q503421) sebagai pemain sepak bola Swiss dan "Alain Sutter" (Q65602988) sebagai

politikus Prancis. Dari kedua entitas tersebut, hanya entitas "Alain Sutter" (Q503421) yang memiliki relasi dengan "position played on team" (P413). Dengan demikian, entitas "Alain Sutter" (Q503421) merupakan entitas yang diperlukan. Falcon 2.0 tidak dapat membedakan entitas yang diperlukan karena Falcon 2.0 tidak dapat memperoleh relasi yang benar. Tabel VIII sampai Tabel X menunjukkan pertanyaan pada tugas *entity linking* yang dapat dijawab dengan pendekatan yang diusulkan tetapi tidak dapat dijawab oleh Falcon 2.0.

V. KESIMPULAN

Pendekatan yang diusulkan yang menggunakan pendekatan pencarian berjenjang dapat digunakan untuk menghilangkan ambiguitas entitas dan mengatasi masalah kesenjangan leksikal pada tugas *entity linking* dan *relation linking* untuk sistem KGQA. Pendekatan yang diusulkan ini dievaluasi dalam SimpleQuestions dan LC-QuAD 2.0 melalui Wikidata. Percobaan menunjukkan bahwa pendekatan yang diusulkan mengungguli Falcon 2.0 pada tugas *entity linking* dan *relation linking* untuk *dataset* SimpleQuestions dan LC-QuAD 2.0. Pendekatan yang diusulkan menghasilkan peningkatan sekitar 40% untuk *entity linking* dan peningkatan 30% untuk *relation linking*.

Pendekatan yang diusulkan tidak secara eksplisit mengatasi masalah relasi tersembunyi. Namun, dengan mempertimbangkan kata tanya yang disebutkan dalam sebuah pertanyaan, penggunaan pendekatan ent-token-question yang menerapkan *universal sentence encoder* dapat memprediksi pasangan hubungan entitas terdekat antara frasa dalam pertanyaan dan *triple* dalam KG.

Penelitian selanjutnya dapat mempertimbangkan penggunaan korpus berbasis teks untuk memperkaya informasi suatu entitas. Korpus digunakan ketika informasi suatu entitas dari KG tidak tersedia secara lengkap. Dengan demikian, tugas menghilangkan ambiguitas entitas dapat bekerja dengan lebih baik dalam memilih entitas yang tepat. Selain itu, penggunaan properti owl:sameAs dapat dipertimbangkan untuk menemukan entitas di KG yang mungkin ditulis dalam leksikal berbeda.

Pendekatan gabungan (*hybrid*) untuk tugas *entity linking* dan *relation linking* untuk sistem KGQA juga dapat menjadi penelitian yang menarik. Penggunaan KG lain untuk mendapatkan sumber daya lain tentang entitas yang terkait dengan pertanyaan tersebut juga menarik untuk dipertimbangkan.

KONFLIK KEPENTINGAN

Penulis menyatakan bahwa tidak terdapat konflik kepentingan.

KONTRIBUSI PENULIS

Konseptualisasi, Adila Alfa Krisnadhi; metodologi, Mohammad Yani; analisis formal, Adila Alfa Krisnadhi; investigasi, Mohammad Yani; penulisan—penyusunan draf asli, Mohammad Yani; penulisan—peninjauan dan penyuntingan, Adila Alfa Krisnadhi dan Indra Budi; validasi, Indra Budi.

UCAPAN TERIMA KASIH

Penelitian ini didanai oleh Fakultas Ilmu Komputer Universitas Indonesia.

REFERENSI

[1] D. Jurafsky dan J.H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, 2nd ed. London, Inggris: Prentice Hall, 2009.

[2] M. Yani, A.A. Krisnadhi, dan I. Budi, "A better entity detection of question for knowledge graph question answering through extracting position-based patterns," *J. Big Data*, vol. 9, hal. 1–26, Jun. 2022, doi: 10.1186/s40537-022-00631-1.

[3] M. Yani dan A.A. Krisnadhi, "Challenges, techniques, and trends of simple knowledge graph question answering: A survey," *Inf.*, vol. 12, no. 7, hal. 1–31, Jul. 2021, doi: 10.3390/info12070271.

[4] E. Prud'hommeaux dan A. Seaborne (2008) "SPARQL query language for RDF," [Online], https://www.w3.org/TR/rdf-sparql-query/, tanggal akses: 15-Jan-2024.

[5] F. Manola, E. Miller, dan B. McBride (2014) "RDF 1.1 primer," [Online], https://www.w3.org/TR/rdf11-primer/, tanggal akses: 15-Jan-2024.

[6] K. Höffner dkk., "Survey on challenges of question answering in the semantic web," *Semantic Web*, vol. 8, no. 6, hal. 895–920, Agu. 2017, doi: 10.3233/SW-160247.

[7] H. Bast dan E. Haussmann, "More accurate question answering on freebase," *CIKM '15, Proc. 24th ACM Int. Conf. Inf. Knowl. Manag.*, 2015, hal. 1431–1440, doi: 10.1145/2806416.2806472.

[8] S. Shin, X. Jin, J. Jung, dan K.-H. Lee, "Predicate constraints-based question answering over knowledge graph," *Inf. Process. Manag.*, vol. 56, no. 3, hal. 445–462, Mei 2019, doi: 10.1016/j.ipm.2018.12.003.

[9] K. Xu, S. Zhang, Y. Feng, dan D. Zhao, "Answering natural language questions via phrasal semantic parsing," *Nat. Lang. Process. Chin. Comput.*, 2014, hal. 333–344, doi: 10.1007/978-3-662-45924-9_30.

[10] A. Delpeuch, "OpenTapioca: Lightweight entity linking for Wikidata," 2019, *arXiv: 1904.09131*.

[11] M. Dubey dkk., "AskNow: A framework for natural language query formalization in SPARQL," *Proc. 13th Int. Conf. Semantic Web. Latest Adv. New Domains*, 2016, hal. 300–316, doi: 10.1007/978-3-319-34129-3_19.

[12] M. Dubey, D. Banerjee, D. Chaudhuri, dan J. Lehmann, "EARL: Joint entity and relation linking for question answering over knowledge graphs," 2018, *arXiv: 1801.03825*.

[13] A. Sakor, K. Singh, dan M.-E. Vidal, "FALCON: An entity and relation linking framework over DBpedia," *Proc. ISWC 2019 Satell. Tracks (Posters Demonstr. Ind. Outrageous Ideas) co-located with 18th Int. Semantic Web Conf. (ISWC 2019)*, 2019, hal. 265–268.

[14] A. Sakor, K. Singh, A. Patel, dan M.-E. Vidal, "Falcon 2.0: An entity and relation linking tool over Wikidata," *CIKM '20, Proc. 29th ACM Int. Conf. Inf. Knowl. Manag.*, 2020, hal. 3141–3148, doi: 10.1145/3340531.3412777.

[15] C. Unger dkk., "Template-based question answering over RDF data," *WWW '12, Proc. 21st Int. Conf. World Wide Web*, 2012, hal. 639–648, doi: 10.1145/2187836.2187923.

[16] K. Xu, S. Zhang, Y. Feng, dan D. Zhao, "Answering natural language questions via phrasal semantic parsing," *Natural Lang. Process. Chin. Comput.*, 2014, hal. 333–344, doi: 10.1007/978-3-662-45924-9_30.

[17] (2024) The Wikimedia website. [Online], https://dumps.wikimedia.org/wikidatawiki/entities/latest-all.nt.gz, tanggal akses: 15-Jan-2024.

[18] J. Devlin, M.-W. Chang, K. Lee, dan K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2018, *arXiv: 1810.04805*.

[19] (2024) The Hugging Face website. [Online], https://huggingface.co/bert-base-cased, tanggal akses: 15-Jan-2024.

[20] T. Wolf dkk., "HuggingFace's transformers: State-of-the-art natural language processing," 2019, *arXiv: 1910.03771*.

[21] D. Cer dkk., "Universal sentence encoder for English," *Proc. 2018 Conf. Empir. Methods Natural Lang. Process., Syst. Demonstr.*, 2018, hal. 169–174, doi: 10.18653/v1/d18-2029.

[22] A. Bordes, N. Usunier, S. Chopra, dan J. Weston, "Large-scale simple question answering with memory networks," 2015, *arXiv: 1506.02075*.

[23] D. Diefenbach, T.P. Tanon, K. Singh, dan P. Maret, "Question answering benchmarks for Wikidata," *Proc. ISWC 2017 Posters Demonstr. Ind. Tracks co-located with 16th Int. Semantic Web Conf. (ISWC 2017)*, 2017, hal. 1–4.

[24] M. Dubey, D. Banerjee, A. Abdelkawi, dan J. Lehmann, "LC-QuAD 2.0: A large dataset for complex question answering over Wikidata and DBpedia," *Semant. Web – ISWC 2019*, 2019, hal. 69–78, doi: 10.1007/978-3-030-30796-7_5.

[25] M. Farda-Sarbas dan C. Müller-Birn, "Wikidata from a research perspective - A systematic mapping study of Wikidata," 2019, *arXiv: 1908.11153*.