AUTOMATIC GENERATING SEMANTICS MARKUP WEBPAGE FOR STRUCTURED DATA

Maysa Marshallia
Informatics, Faculty of Mathematic and Natural Science
Universitas Sebelas Maret
maysamarshallia@student.uns.ac.id

Dewi Wisnu Wardani
Informatics, Faculty of Mathematic and Natural Science
Universitas Sebelas Maret
dww_ok@staff.uns.ac.id

ABSTRACT
Along with the development of knowledge graph, required larger structured data to build it. Publishing large structured data required greater effort. In this paper will be discussed how to create a framework to create an automatic semantic annotation for relational database. By using this framework no matter how much data to be marked it will be faster and required less effort. We purpose the use of semantic similarity to provide semantic markup based on schema.org. Wu Palmer Similarity and Wordnet Synsets will be used to calcite the similarity between table's atribut and class's property. The calculation will yield a candidate property to be used in semantics markup; thereafter, a webpage that contain those markups will be generated. The last step of the purposed method is generating knowledge graph. The result of this paper shown that using Wordnet Synsets give better result than only Wu Palmer Similarity.

Keywords
Semantic annotation, semantic markup, knowledge graph, schema.org, schematic authoring

1. PENDAHULUAN

Knowledge graph adalah *knowledge base* dari fakta yang merepresentasikan entitas di dunia nyata seperti orang atau organisasi [1]. Dalam *search engine, knowledge graph* memberikan *recommendation, ranking, web searching,* dan *exploratory search,* dengan mengumpulkan informasi dari entitas dan *link* dari berbagai sumber, pada saat yang sama menyediakan *property* dan tipe dari entitas[2].Google Knowledge Graph, Amazon Product Graph, Facebook Graph API, IBM Watson, dan Microsoft Satori adalah beberapa produk yang menggunakan *knowledge graph*.

Terdapat dua cara untuk memperoleh *entities* dari sebuah laman yaitu, secara implisit (dengan menggunakan NLP *Natural Language Processing*) dan secara eksplisit (menggunaan *structured data) [3]*. *Structured data* adalah standar formal untuk menyediakan informasi tentang laman web dan klasifikasi dari isi laman tersebut[4]. Proses memberikan *markup structured data* disebut dengan *semantic annotation*. *Semantic annotation* memperkaya *content* dari sebuah laman dengan informasi yang dapat dipahami oleh mesin[5].

Pada tahun 2011 google, yandex, Microsoft, dan yahoo memperkenalkan schema.org sebagai vocabulary yang digunakan untuk menerbitkan *structured data*, dapat digunakan menggunakan format RDFa, Microdata, dan JSON-LD [6]. Dari ketiga format yang telah disebutkan, JSON-LD merupakan format yang direkomendasikan oleh Google [4] .

S*emantic annotation* dapat dilakukan secara manual maupun secara otomatis. *Markup* yang diberikan secara manual tentunya akan lebih baik dari pada *markup* yang diberikan secara otomatis. Hal ini dikarena pengguna dapat memilih property yang paling sesuai dengan kebutuhannya.

Pada penelitian sebelumnya [7] *semantic annotation* dilakukan secara manual. Di mana pengguna memilih domain yang diingikan kemudian menginputkan nilai untuk semua *property* yang disediakan dari domain yang telah dipilih. Hal tersebut dilakukan untuk menghindari ketidaklengkapan data dan menghidari kesalahan pengguna. *Manual annotation* digunakan untuk memberi *markup unstructured data* seperti pada penelitian [8]

Dari latar belakang di atas maka pada penelitian ini dilakukan pendekatan yang berbeda dari penelitan [7]. Pada penelitian ini akan dibuat *framework* yang dapat melakukan *automatic semantic annotation* pada data terstruktur. Di mana nama tabel akan dipilih sebagai domain dari *markup* dan nama atribut akan di-*matching* dengan properti yang dimiliki oleh domain untuk dijadikan *property* pada *markup*. Sedangkan nilai dari *property* akan diambil dari *record* tabel.

2. PENELITIAN TERKAIT

Pada penelitian sebelumnya [11] digunakan metode a*utomatic semantic annotation* untuk memberikan *markup* pada BIM (*Building Information Modelling)* berdasarkan *ontology IFC*. Sistem yang dibuat memungkinkan s*emantic annotation* dilakukan dalam level dokumen maupun *term,* selain itu sistem juga mampu melakukan pencarian secara semantik.Untuk memberikan *markup* pada level kata Gao menggunakan Wordnet Synsets untuk mencari kandidat *term* dari *ontology* yang telah dibuat. Wordnet Synsets akan menghasilkan sinonim konsep yang ada dalam *ontology*. Selain menggunakan Wordnet Synsets dapat pula digunakan metode lain untuk mencari kemiripan antara dua *term*.

Penelitian [12] menggunakan *wordnet similarity* dan *cosine similarity* sebagai metode untuk mengukur keefektivan pemetaan r*elational database* ke dalam *graph model*. Pada penelitian tersebut r*elational database* dipetakan menjadi g*raph models* tanpa kehilangan aspek semantiknya dengan menggunakan schema.org sebagai *vocabulary*. Pada penelitian [1] Pawar dan Moga mengusulkan algoritma untuk mencari nilai kemiripan kalimat dengan menggunakan statistic dan *corpus-based*. Penelitian tersebut mengdapatkan nilai korelasi manusia sebesar 0.8794.

3. METODOLOGI PENELITIAN

Dalam penelitian ini terdapat beberapa tahapan penelitian yaitu, *data gathering, word semantic similarity, generate markup, generate webpage,* dan *generate knowledge base*.

3.1. Data Gathering

Data yang digunakan pada penelitian ini berasal dari schema.org dan universitas sebelas maret. Data yang diambil dari schema.org terdiri dari *class*, *property*, hierarki, dan tipe yang diharapkan sebagai *value* untuk setiap *property*. Data diberikan marukup berasal dari Universitas Sebelas Maret. Tabel yang digunakan bersifat *single table*.

3.2. Pembentukan Algoritma

Setelah semua data didapatkan, tahap selanjutnya adalah pembentukan algoritma. Algoritma yang diusulkan memiliki beberapa tahapan didalamnya. Berikut ini tahapan dalam algoritma yang diusulkan;

Purposed Algorithm

Input : Data *attribute table , property class schema.org*
Output : Data nilai kemiripan *attribute* dan *property*

Proses :
Begin
list attribute;
list property;
string markup;
list result[int score, string attribute, string property];
list helper[int score, string attribute, string property];
//Preprocessing data
property<- Tokenization(property);
property<- removeStopWord(property);
property <- toLowerCase(property);
property <- POS-tagging(property);
attribute<- Tokenization(attribute);
attribute<- removeStopWord(attribute);
attribute<- toLowerCase(attribute);
attribute<- POS-tagging(attribute);

```
while(attribute){
    while(property){
        score=wup_similarity(attribute,property):
        if attribute is in wordnet_synsets(property){
            score+=1;
        }
        Score=score/2;
        helper.append(score, property, attribute);

    }
    result.append(max(helper sort by score));
}
```

Gambar 1. Gambar algoritma *similarity*

3.2.1. *Word Semantic Similarity*

Karena data yang akan diberikan markup menggunakan bahasa Indonesia sedangkan schema.org menggunakan bahasa Inggris, sehingga perlu ditranslasi. Proses translasi dilakukan menggunakan Google Translate API.
Setelah semua atribut ditranslasi ke dalam bahasa Inggris, langkah selanjutnya adalah *word similarity matching*. Untuk mencari *property* yang paling mirip dengan atribut tabel digunakan metode Wu Palmer Similarity dan Wordnet Synsets.

3.2.2. *Wu Palmer Similarity*

Pada tahun 1994 Wu dan Palmer mengenalkan sebuah metode untuk mengukur kemiripan antar konsep berdasarkan panjang path, LCS (*Least Common Subsumer*), dan jarak *node* ke *root* [13]. Wu Palmer *similarity* dapat diukur menggunakan rumus

$$\delta wuPalmer(C_p, C_q) = \frac{(2d)}{(L_p + L_q + 2d)} \qquad (1)$$

Lp = jarak node Cp ke node LCS
Lq = jarak node Cq ke node LCS
d = jarak node LCS ke root

3.2.3. *Wordnet Synsets*

WordNet® adalah sebuah kamus besar bahasa Inggris. Kata benda, kata kerja, kata sifat dan keterangan dikumpulkan ke dalam kumpulan set sinonim yang kognitif (*synsets*), setiap *synsets* mengekspresikan konsep yang berbeda. Wordnet mengumpulkan kata dalam bahasa Inggris menjadi kumpulan yang disebut dengan *synsets*, unit terkecil dari Wordnet [14]. Ketika sebuah kata ditemukan dalam beberapa *synsets*, maka hubungan antara kata-kata tersebut disebut dengan polisemi. Synsets ditampilkan berdasarkan urutan frekuensi kemunculan [3].

Kedua metode tersebut digabungkan menggunakan menggunakan formula berikut,

$$similarity(a, b) = \frac{(\delta wuPalmer + k)}{2} \qquad (2)$$

Apabila atribut ditemukan sebagai salah sinonim dari *property* maka **k** akan bernilai 1, jika tidak maka k=0. Pasangan atribut dan *property* yang memiliki nilai tertinggi akan dipilih sebagai kandidat untuk *markup*. Rumus diatas hanya digunakan untuk mencari nilai kemiripan antar dua kata, nilai kemiripan antar atribut dan property didapatkan melalui tahap normalisasi.

3.2.4. Normalisasi

Pada tahap ini akan dilakukan normalisasi untuk mencari kemiripan antara atribut dan property menggunakan algoritma yang terdapat pada penelitian Pawar and Mago[1]. Input dari algoritma diatas adalah list yang berisi hasil tagging atribut (S1) dan property (S2). Nilai kemiripan dari S1 terhadap S2 akan disimpan dalam vector V1, nilai kemiripan S2 terhadap akan disimpan dalam vector V2. Untuk mencari nilai kemiripan(S) dari vector V1 dan V2 digunakan dot product. Kata yang miliki kemiripan paling memiliki efek besar pada nilai vector, sehingga digunakan ζ. Nilai ζ didapatkan dari jumlah nilai valid dalam vector V1 dan V2 dibagi γ. Menurut Rubinstein 1965 dua term dianggap sinonim jika nilai kemiripan nya lebih dari 0.8025. Nilai γ ditetapkan sebesar 1.8.

3.2.5. *Generate Markup*

Hasil dari *semantic similarity* akan dijadikan bahan untuk *markup* laman web. Di mana domain yang dipilih menjadi tipe dari anotasi dan *property* akan menjadi *property markup* yang nilainya diambil dari tabel. Pasangan atribut dan *property* dibuat markup structured data dengan format JSON-LD. JSON-LD dipilih karena memenuhi kriteria berikut [15];

- *simplicity*: tidak diperlukan *library* tambahan untuk membuatnya, mudah dipelajari, *developers* hanya perlu mempelajari JSON dan dua *keywords* untuk menggunakan fungsi dasar dalam JSON-LD.

- *Compatibility* : JSON-LD adalah JSON yang *valid*, semua standar yang dimiliki JSON dokumen berjalan dengan baik di JSON-LD.
- *Expressivess* : *syntax* yang digunakan merepresentasikan directed graph , hal ini memastikan hampir semua data model dunia nyata dapat diekspresikan.
- *Terseness* : *syntax* JSON-Ld sangat ringkas dan mudah dibaca manusia .

Algoritma Normalisasi
Input: list S1, list S2
Output : score
Process:
S1<- list of tagged tokens
S2<-list of tagged tokens
Vector_length<-max(len(S1),len(S2))
V1,V2<- vector_length(null)
V1,V2<- vector length(word_similarity(S1,S2))
ζ = 0
while S1 list of tagged tokens **do**
    if word similarity value > benchmark_similarity_value **then**
    C1 ←C1+1
while S2 list of tagged tokens **do**
    if word similarity value > benchmark_similarity_value **then**
    C2 ←C2+1
ζ ← sum(C1,C2)/γ
S<-||V1||.||V2||
If sum(C1,C2)=0 **then**
ζ=<-vector_length/2
Score<-S/ ζ

Gambar 2. Gambar algoritma normalisasi

3.3.Generate Webpage

Data dari tabel yang ingin diberikan akan dibuat menjadi laman web dengan template yang telah dibuat sebelumnya. Kemudian akan ditambahkan *markup* dari proses sebelumnya pada bagian *head* laman web.

3.4.Create Knowledge Graph

Pada tahapan ini *markup* dan isi dari laman web akan di-*index* dengan menggunakan *spider*. *Markup* yang diambil dijadikan *knowledge graph*. *Property* yang dijumpai pada *markup* akan menjadi *edge* dari graf, sedangkan nilai dari *property* akan digunakan sebagai *node*.

4. EXPERIMENTAL

Tahap *experimental* terdiri dari tiga bagian, *experimental environment, result,* dan *result analysis.*

4.1.Experimental Environment

Tabel 1. Tabel *experimental environment*

Penggunaan	**Program/API**
Data Gathering	Spider
Translation	Google Translation API
Pembentukan Algoritma	Python2.7
Database	MongoDB
Operating System	Ubuntu

4.2.Result

Bagian *result* akan dijelaskan mengenai hasil dari percobaan yang telah dilaksanakan. Setiap tahapan yang ada pada metodologi penelitian akan dijabarkan hasilnya pada bagian ini.

4.2.1. *Data Gathering*

Data diambil dari schema.org melalui proses *crawling*. Data yang diambil berupa nama class beserta *property* dan *expected type* untuk setiap *property*. Karena memiliki beberapa *expected type* sebagai nilai dari *property*, sehingga memungkinkan untuk memiliki nested value. Berikut ini beberapa property dari *class Person*. Selain data tersebut, diambil pula hierarki dari *class* yang ada dalam schema.org. Setiap *class* akan meng-inheritance property dari class parent. *Class Person* merupakan *child* dari *class Thing*, sehingga *property class Thing* dimiliki oleh *class Person*. Dari hasil *crawling* schema.org didapatkan 604 class beserta *property* dan hierarkinya.

Tabel 2. Tabel *property class Person* dari schema.org

Property	*Expected Type*
AdditionalName	*Text*
jobTitle	*Text*
address	*PostalAddress, Text*
birthDate	*Date*
birthPlace	*Place*
gender	*GenderType, Text*

Sedangkan data yang akan diberikan *markup* adalah data yang berasal dari *single table*. Atribut dari tabel tersebut akan digunakan untuk menentukan *property* mana yang akan digunakan untuk *markup*. Berikut ini hasil translasi dari atribut dari tabel yang akan diberikan *markup*,

Tabel 3. Tabel atribut yang digunakan

ID	*religion*
Name	*lastEducation*
Sex	*jobTitle*
placeOfBirth	*faculty*
dateOfBirth	*department*
haveAJob	

4.2.2. *Purposed Algorithm*

4.2.2.1. *Prepocessing*

Data yang telah diperoleh dari schema.org maupun data yang akan diberikan *markup* akan diproses terlebih dahulu. Berikut ini langkah yang digunakan;

4.2.2.1.1.Tokenization

Untuk penulisan pemisahan perkata pada schema.org digunakan huruf kapital. Seperti *property additionalName* pada *class Person*, kata *additional* dan kata *name* digabungkan dengan menuliskan huruf N dengan kapital. Penulisan nama atribut pada tabel dapat menggunakan format seperti *property class* ataupun menggunakan pemisah seperti '_' dan '-'. Untuk memisahkan *term* tersebut maka digunakan *regex*. Berikut ini contoh hasil tokenization untuk atribut haveAJob,

Tabel 4. Tabel hasil tonenization

Before	After
'haveAJob'	['have','A','Job']

4.2.2.1.2.Removing Stopwords

Pada tahap ini *stopwords* akan dihilangkan dari atribut. *Stopwords* yang dihilangkan seperti *a,this, that*, dan lainnya. Pada atribut *haveAJob* terdapat *stopword a* dan *have* sehingga perlu dihilangkan.

Tabel 5. Tabel hasil *removing stop words*

Before	After
['Job']	['Job']

4.2.2.1.3.Parts of Speech Tagging

Pada tahap ini tiap *kata* pada atribut dan *property* akan diberikan tag sesuai jenis kata. Jenis kata yang digunakan adalah kata sifat (*adjective*), kata kerja (*verb*), kata benda (*noun*), dan kata keterangan (*adverb*). Berikut ini hasil tagging untuk atribut *haveAJob*,

Tabel 6. Tabel hasil *parts of speech tagging*

Before	*After*
['Job']	[('Job','N')]

4.2.2.1.4.Mengubah menjadi lower case

Setelah *property* dan atribut diberikan *tagging* selanjutnya atribut dan *property* akan diubah menjadi *lowercase*.

Tabel 7. Table hasil mengubah ke *lowercase*

Before	After
[('Job','N')]	*[('job','N')]*

4.2.2.2.*Semantic Similarity Matching*

Pada tahap ini atribut tabel akan dihitung nilai kemiripannya dengan *property* dari *class* yang sebelumnya telah dipilih oleh pengguna. Berikut ini hasil perhitungan untuk atribut table dan *property*.

```
<script type="application/ld+json">{
"@context": "http://schema.org",
"@type": "Person",
"identifier":"1005",
"name":"Prof.Dr. RAVIK KARSIDI, M.S.",
"gender":"Laki Laki",
"birthPlace":{
    "@type":"Place",
    "name":" SRAGEN"},
"birthDate":{
    "@type":"Date",
    "name":"1957-07-07"},
"jobTitle":"Staf Pendidik PNS",
"known":"Fakultas KIP",
"familyName":"S-1 Pendidikan Luar Biasa",
"worksFor":{
"@type":"Organization",
"name":"IV/d"},
"hasOccupation":{
    "@type":"Occupation",
    "name":"PEMBINA UTAMA MADYA"}
    }
</script>
```

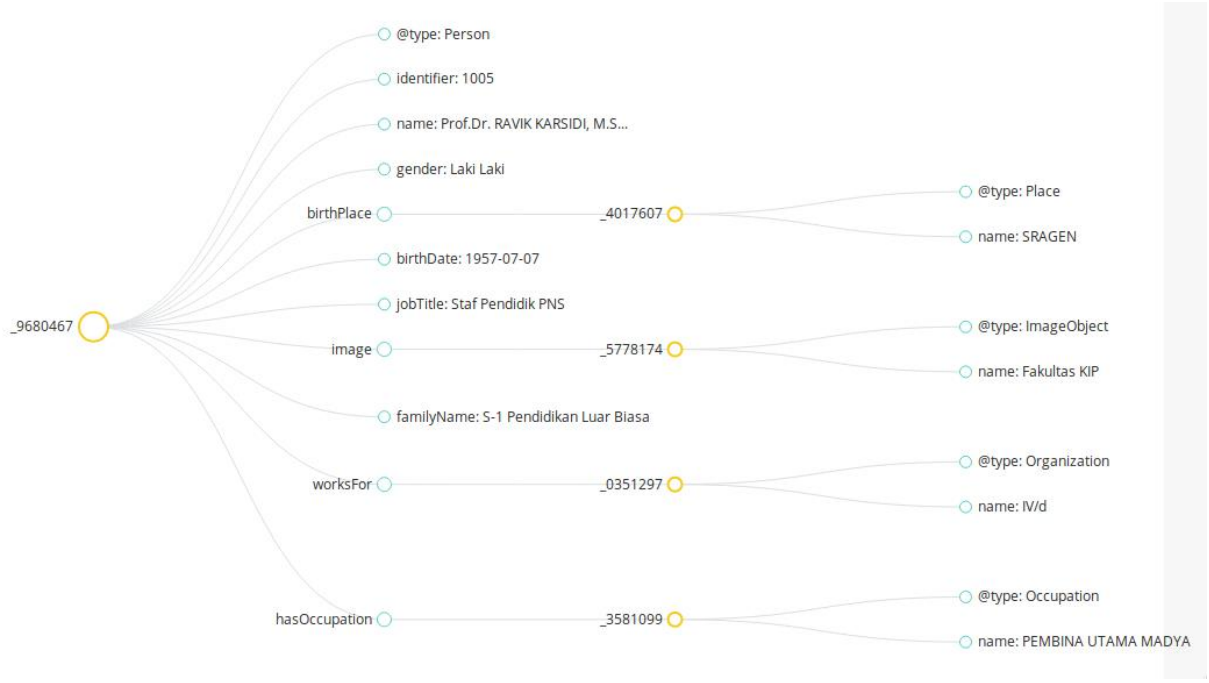
Gambar 3. Gambar hasil *markup*

Tabel 8. Tabel hasil perhitungan dengan wup dan wup+k

Atribut	*Property*	wup	wup+k
Identifier	*identifier*	0.899999	1.0
Name	*name*	0.899999	1.0
Gender	*gender*	0.899999	0.895835
PlaceOfBirth	*birthPlace*	0.899999	0.9375
DateOfBirth	*birthDate*	0.899999	0.35714
Religion	*image*	1.020408	0.87053
Education	*jobTitle*	0.793577	0.90441

Requirement			
JobTitle	*HasOccupation*	0.899999	0.61538
Faculty	*knows*	0.653061224	0.375
Department	*familyName*	0.5625	0.35294
Classroom	*worksFor*	0.996539	1.0
HaveAJob	*jobTitle*	0.899999	0.81665

Pada hasil perhitungan kemiripan dengan menggunakan rumus 2, *Property image* dan *property hasOccupation* muncul sebanyak dua kali (*faculty-image, religion-image, jobTitle-hasOccupation,* dan *haveAJob-hasOccupation*), oleh karena itu pasangan atribut dan *property* yang miliki nilai kemiripan lebih kecil akan menggunakan *property* yang mimiliki nilai kemiripan tertinggi dibawahnya. *Property* dari atribut *faculty* diganti dengan *property knows* (wup=0.653061224, wup+k=0.1632653).



Gambar 4. Gambar visualisasi makrup dalam graph

4.2.3. *Generate Markup*

Dari hasil *semantic similarity* yang telah didapatkan *property* yang merepresentasikan atribut pada tabel. *Property* tersebut akan diberikan nilai dari *record* yang ada pada tabel. Hasil *markup ditampilkan dalam gamber 3.*

4.2.4. *Generate Webpage*

Markup yang telah dibuat ditambahkan pada laman web yang telah dibuat. *Markup* tersebut ditambahkan pada bagian *head* file html.

4.2.5. *Create Knowledge Graph*

Sebelum dibuat *knowledge graph* laman web yang telah ditambahkan *markup* akan di-*index* terlebih dahulu. Pada saat *indexing*, *markup* dan url akan disimpan.

4.2.6. *Term Searching*

Setelah proses *indexing* dan pembuat laman web selesai dilaksanakan, *term* yang terdapat di dalamnya sudah dapat dicari menggunakan laman yang disediakan.

Dari gambar 6 dapat dilihat dari pencarian pada *term* KLATEN. Pada bagian pada bagian kiri akan ditampilkan markup dari webpage yang mimiliki term yang dicari. Pada bagian kanan akan ditampilkan node yang adjacent dengan node term yang dicari.

4.3.Analisa Hasil

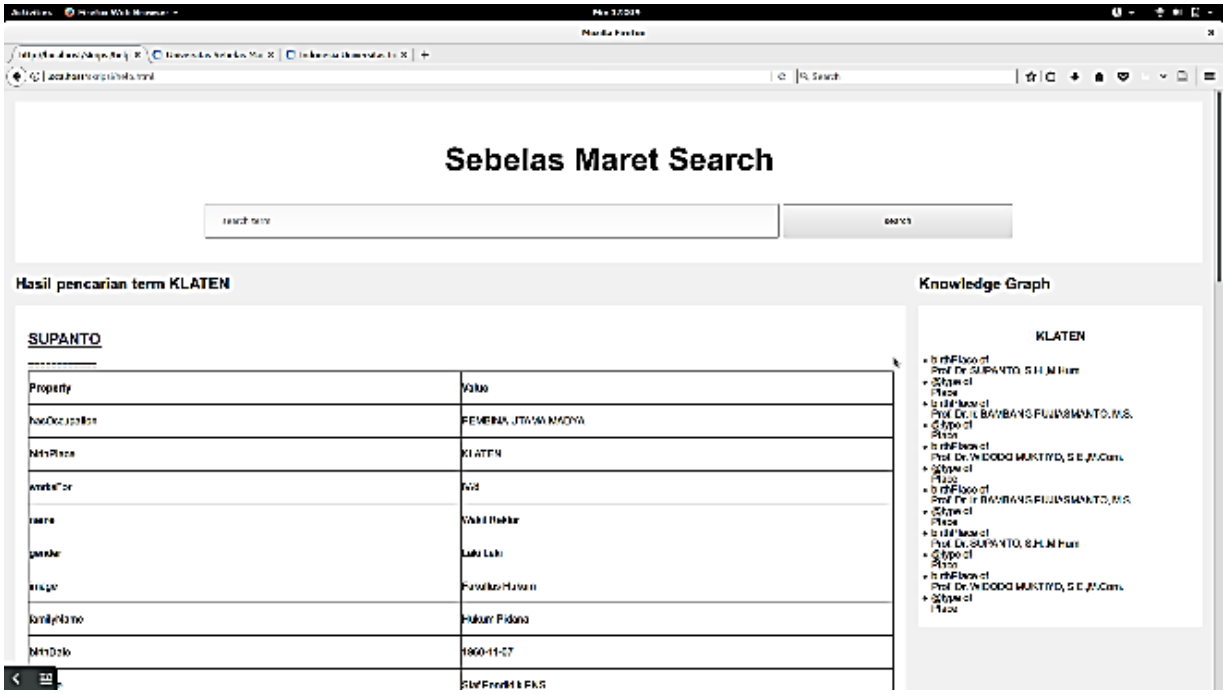
Dari hasil di atas dapat disimpulkan bahwa metode yang diusulkan mendapatkan hasil yang lebih baik jika dibandingkan hanya menggunakan Wu Palmer *similarity*.

Atribut seperti golongan ruang, fakultas, dan unit tidak memenuhi apa yang diharapkan oleh peneliti. Hal ini dikarenakan beberapa hal, pertama hasil translasi yang tidak sesuai karena beberapa istilah hanya digunakan di Indonesia. Istilah seperti golongan ruang ditranslasikan sebagai *classroom*, yang secara semantik sangat berbeda makna nya dengan golongan ruang. Kedua, beberapa atribut tidak ada dalam *property class* yang terpilih.

5. KESIMPULAN

Dari penelitian di atas telah membuat framework untuk melakukan *automatic semantic annotation* untuk data terstruktur. *Semantic markup* dibuat dengan menggunakan schema.org yang di-*matching* dengan atribut tabel. Metode yang diusulkan untuk semantic matching adalah Wu Palmer Similarity dan Wordnet Synsets. Wu Palmer digunakan untuk mencari kemiripan antara dua *term* berdasarkan hierarkinya. Sedangkan

Wordnet Synsets digunakan untuk mencari kemiripan dua *term* secara semantik.



Gambar 5. Gambar hasil pencarian term

Hasil percobaan menunjukkan bahwa penggunaan Wordnet Synsets memberikan hasil yang lebih baik dari pada hanya menggunakan Wu Palmer Similarity. *Framework* yang diusulkan memungkinkan pengguna untuk memberikan *semantic markup* secara otomatis pada data terstruktur. Hasil dari penelitian ini diterapkan dalam *search engine* untuk menunjukan penggunaan *knowledge graph* yang dibangun dari *semantic markup* yang telah dibuat. Pada penelitian selanjutnya dapat dibuat framework yang dapat mengenerate laman web untuk data terstruktur yang bersifat *multi table*.

6. DAFTAR PUSTAKA

[1] M. Rospocher *et al.*, "Building event-centric knowledge graphs from news," *Web Semant. Sci. Serv. Agents World Wide Web*, vol. 37, pp. 132–151, 2016.

[2] N. Voskarides, E. Meij, M. Tsagkias, M. De Rijke, and W. Weerkamp, "Learning to explain entity relationships in knowledge graphs," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 2015, vol. 1, pp. 564–574.

[3] K.-T. Sun, H. Yueh-Min, and L. Ming-Chi, "A WordNet-based near-synonyms and similar-looking word learning system," *J. Educ. Technol. Soc.*, vol. 14, no. 1, p. 121, 2011.

[4] "Structured Data General Guidelines | Search," *Google Developers*. [Online]. Available: https://developers.google.com/search/docs/guides/sd-policies. [Accessed: 23-Jun-2018].

[5] "Demystifying The Google Knowledge Graph," *Search Engine Land*, 02-Sep-2014. [Online]. Available: https://searchengineland.com/demystifying-knowledge-graph-201976. [Accessed: 23-Jun-2018].

[6] "Home - schema.org." [Online]. Available: https://schema.org/. [Accessed: 20-Jun-2018].

[7] U. Şimşek, E. Kärle, O. Holzknecht, and D. Fensel, "Domain specific semantic validation of schema. org annotations," in *International Andrei Ershov Memorial Conference on Perspectives of System Informatics*, 2017, pp. 417–429.

[8] A. Khalili and S. Auer, "Wysiwym authoring of structured content based on schema. org," in *International Conference on Web Information Systems Engineering*, 2013, pp. 425–438.

[9] Z. Akbar, E. Kärle, O. Panasiuk, U. Şimşek, I. Toma, and D. Fensel, "Complete Semantics to empower Touristic Service Providers," in *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, 2017, pp. 353–370.

[10] E. Kärle, U. Şimşek, and D. Fensel, “semantify. it, a platform for creation, publication and distribution of semantic annotations,” *ArXiv Prepr. ArXiv170610067*, 2017.

[11] G. Gao, Y.-S. Liu, P. Lin, M. Wang, M. Gu, and J.-H. Yong, “BIMTag: Concept-based automatic semantic annotation of online BIM product resources,” *Adv. Eng. Inform.*, vol. 31, pp. 48–61, 2017.

[12] D. W. Wardani and Josef Küng, “THE EFFECTIVENESS OF THE SEMANTIC MAPPING RELATIONAL TO GRAPH MODEL,” *Iadis Appl. Comput. 2016*, pp. 107–114, 2016.

[13] T. Wei, Y. Lu, H. Chang, Q. Zhou, and X. Bao, “A semantic approach for text clustering using WordNet and lexical chains,” *Expert Syst. Appl.*, vol. 42, no. 4, pp. 2264–2275, 2015.

[14] “JSON-LD 1.0.” [Online]. Available: https://www.w3.org/TR/2014/REC-json-ld-20140116/. [Accessed: 25-Jun-2018].