Pembentukan Tesaurus pada Cross-Lingual Text dengan Pendekatan Constraint Satisfaction Problem

Umy Chasanah Noor Rizqi, Chastine Fatichah, Diana Purwitasari
Departemen Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember (ITS)
Jl. Arief Rahman Hakim, Surabaya 60111 Indonesia
e-mail: chastine@if.its.ac.id, diana@if.its.ac.id, umy.rizqi13@mhs.if.its.ac.id

***Abstrak*—Dokumen studi dan tesis sering kali disediakan dalam dua bahasa, yaitu bahasa Indonesia dan Inggris. Dalam pencarian, setiap mahasiswa memiliki kecenderungan mencari dokumen dengan menggunakan kata kunci dengan bahasa tertentu. Tujuan dari penelitian ini adalah untuk membangun *cross-lingual* tesaurus bahasa Indonesia dan bahasa Inggris dengan pendekatan *Constraint Satisfaction Problem*. Dalam penelitian ini digunakan data Studi serta Tesis mahasiswa Institut Teknologi Sepuluh Nopember. Pada pengolahan dokumen dilakukan beberapa langkah yaitu pembentukan *pararell corpus*, ekstraksi kata, pembobotan kata, dan pembentukan informasi *co-occurrence*, yang selanjutnya dilakukan *Constraint Satisfaction Problem* dengan *backtracking* sebagai solusi pencarian. Pembobotan menggunakan TF-IDF (*term frequency–inverse document frequency*). Hasil dari proses pembangunan tesaurus, tesaurus yang dibentuk dengan menggunakan CSP menghasilkan *precision* 91,38% sedangkan tesaurus yang dibentuk tanpa menggunakan CSP menghasilkan *precision* 45,23%. Pencarian dokumen menggunakan tesaurus menghasilkan *recall* 86,67%, *precision* 100% dan akurasi 86,67%.**

Kata Kunci—Backtracking, Co-occurrence, Constraint Satisfaction Problem , Cross-lingual, Tesaurus.

I. PENDAHULUAN

PENCARIAN dokumen adalah hal yang esensial dalam bidang *text mining*. Dokumen studi dan tesis sering kali disediakan dalam dua bahasa, yaitu bahasa Indonesia dan Inggris. Dalam pencarian, setiap mahasiswa memiliki kecenderungan mencari dokumen dengan menggunakan kata kunci dalam bahasa tertentu.

Cross-lingual tesaurus adalah padanan kata dari suatu kalimat pada bahasa tertentu ke dalam dua bahasa, yaitu bahasa Indonesia dan bahasa Inggris. *Cross-lingual* tesaurus yang terbentuk bisa sebagai sinonim, terjemahan kata maupun akronim[1][2].

Pada penelitian Bel et al[3] *cross-lingual* tesaurus digunakan untuk terjemahan. Pembentukan tesaurus digunakan untuk *query expansion* pada pencarian dokumen. *Query expansion* merupakan proses merumuskan *query* asli untuk meningkatkan kinerja pengambilan dalam operasi pencarian informasi. Dalam konteks mesin pencari, *query expansion* melakukan evaluasi masukan pengguna dan memperluas permintaan pencarian untuk mencocokkan dokumen tambahan.

Pembentukan *Cross-lingual* tesaurus, membutuhkan *parallel corpus*, *parallel corpus* adalah kumpulan dari pasangan dokumen dalam dua bahasa (bahasa Indonesa dan bahasa Inggris), dimana pasangan dokumen adalah terjemahan dari antar kedua dokumen. Sebuah *parallel corpus* adalah kumpulan dokumen dalam bahasa masing-masing dan dikombinasikan atas dasar kesamaan konten pada judul.

Constraint Satisfaction Problem merupakan salah satu metode yang dapat dipergunakan untuk pembentukan *cross lingual* tesaurus dengan mempertimbangkan *constraint* sehingga diharapkan kata yang dihasilkan memiliki keterkaitan yang lebih baik. Untuk memenuhi *constraint satisfaction problem* dibutuhkan suatu metode pencarian, salah satunya adalah *backtracking*. *Backtracking* adalah algoritma untuk *mencari* solusi persoalan secara lebih ringkas dan juga merupakan perbaikan dari algoritma *brute-force*.

Diskusi pada penelitian ini dibagi dalam struktur sebagai berikut. Bab II membahas metodologi yang digunakan dalam penelitian ini. Bab III membahas hasil dan diskusi percobaan yang dilakukan. Terakhir pada Bab IV membahas kesimpulan dari hasil yang didapatkan dalam penelitian ini.

II. METODOLOGI

Gambaran umum alur dan metode yang digunakan dalam penelitian ini ditujukan pada Gambar 1.

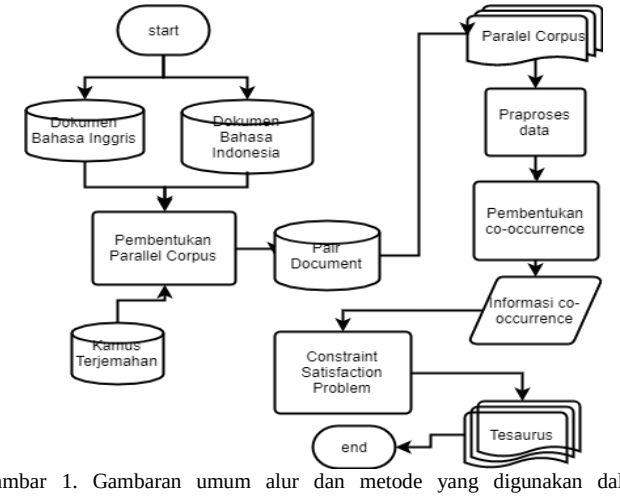
A. *Pembentukan Parallel Corpus*

Pengumpulan *parallel corpus* dilakukan dengan melakukan penterjemahan dokumen berbahasa Inggris ke bahasa Indonesia, kemudian dihitung nilai jarak antar dokumen terjemahan dan dokumen bahasa Indonesia menggunakan *Jaccard similarity*. Selanjutnya diambil dokumen bahasa Inggris dengan nilai kesamaan tertinggi dari setiap dokumen bahasa Indonesia, yang selanjutnya pasangan tersebut disebut *parallel corpus.*

1) Penterjemahan Dokumen

Penterjemahan dokumen dilakukan dua kali yaitu dengan menggunakan pustaka Python *Urllib2* dan *database* kamus bahasa Inggris – Indonesia. *Urllib2* mendukung pengambilan URL untuk banyak "skema URL" (diidentifikasi oleh string sebelum ":" di URL[4]. *Urllib2* digunakan untuk mengakses

"http:// translate.google.com" yang akan digunakan untuk melakukan penterjemahan kata bahasa Inggris kedalam bahasa Indonesia. Penterjemahan juga dilakukan dengan melakukan pencocokan kata bahasa Inggris kedalam *database* kamus bahasa Inggris – Indonesia, sehingga akan didapatkan beberapa kemungkinan kata terjemahan bahasa Indonesia dari kata bahasa Inggris.



Gambar 1. Gambaran umum alur dan metode yang digunakan dalam penelitian ini.

2) *Jaccard Similarity*

Jaccard similarity adalah statistik yang digunakan untuk membandingkan kesamaan sampel set. *Jaccard* koefisien didefinisikan sebagai ukuran potongan (*intersection*) dibagi dengan gabungan (*union*) set sampel. Rumus Penghitungan *Jaccard similarity* ditunjukan pada (1).

$$sim_{jacc}(t_1, t_2) = \frac{|t_1 \cap t_2|}{|t_1 \cup t_2|} \quad (1)$$

B. Algoritma Praproses Teks

Algoritma pemrosesan teks digunakan untuk melakukan ekstraksi kata pada kumpulan dokumen dan pembentukan co-occurrence dari kata hasil ekstraksi.

1) *Tokenisasi dan Case Folding*

Tokenisasi merupakan proses pengambilan kata-kata dari sebuah dokumen yang dipisahkan berdasarkan spasi. Pada proses tokenisasi, yang dijadikan acuan pemisah antar kata adalah tanda baca dan spasi. Contoh pustaka tokenisasi adalah NLTK Tokenizer. Case folding bertujuan untuk mengkonversi keseluruhan teks dalam dokumen menjadi bentuk standar (huruf kecil atau lowercase).

2) *Penghapusan Stopword*

Setelah dilakukan proses tokenisasi pada judul dokumen, proses selanjutnya yaitu penghapusan *stopword*. *Stopword* adalah kata-kata sering muncul pada suatu dokumen yang tidak memberikan informasi penting, seperti kata penghubung dan kata ganti orang. Penghapusan *stopword* ini bertujuan agar kata-kata yang digunakan hanya kata yang memiliki arti penting dan memberikan suatu informasi.

3) *Stemming*

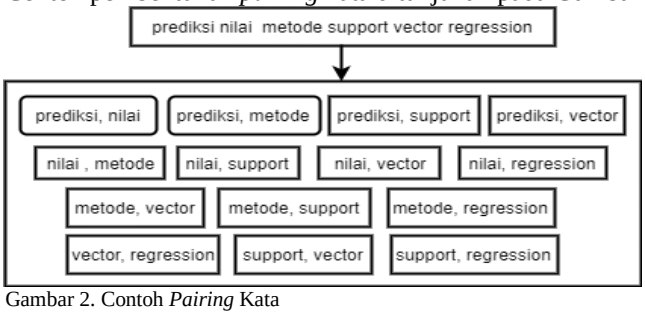
Stemming merupakan suatu proses untuk menemukan kata dasar dari sebuah kata. Proses *stemming* dilakukan dengan menghilangkan semua imbuhan (*afiks*) baik yang terdiri dari awalan (*prefiks*) sisipan (*infiks*) maupun akhiran (*sufiks*) dan kombinasi dari awalan dan akhiran (*konfiks*)[4]. Pustaka Python yang bisa dipakai saat ini untuk bahasa Indonesia adalah Sastrawi dan untuk bahasa Inggris adalah NLTK Snowball.

C. Pembentukan Informasi Co-occurrence

Proses ini membentuk kombinasi kata-kata dari hasil praproses data dari pasangan dokumen pada proses sebelumnya. Kombinasi kata-kata ini nantinya akan digunakan untuk membentuk informasi *co-occurrence* dengan nilai *cluster weight*.

1) *Pairing Kata*

Proses ini membentuk kombinasi kata-kata dari hasil pada proses sebelumnya. Kombinasi kata-kata ini nantinya akan digunakan untuk membentuk informasi *co-occurrence*. Contoh pembentukan *pairing* kata ditunjukan pada Gambar 2.



Gambar 2. Contoh *Pairing* Kata

2) *Pembobotan Term*

Tahap selanjutnya hasil pembentukan *term* dari proses sebelumnya akan dihitung nilai bobot tiap pasangan kata yang nantinya akan digunakan untuk perhitungan *cluster weight.* Perhitungan bobot yang digunakan adalah tf-idf. Rumus perhitungan bobot *term* terdapat pada (2). Sedangkan rumus perhitungan bobot *pairing term* terdapat pada (3).

$$d_{ij} = tf_{ij} \times \log\left(\frac{N}{df_j}\right) \quad (2)$$

$$d_{kij} = tf_{kij} \times \log\left(\frac{N}{df_{ij}}\right) \quad (3)$$

dimana:

tf_{kij} = *term frequency* minimal dari *term i* dan *term frequency term j* pada pasangan dokumen *k*

df_{ij} = *document frequency* dari *term j* dan *term k*.

3) *Cluster Weight*

Cluster Weight digunakan untuk mengukur kedekatan antar kata dalam pasangan kata yang dibentuk[5][6]. Penghitungan *cluster weights* antara *term j* dan *term k* sesuai (4**)** dan perhitungan *weighing factor* sesuai dalam (5).

$$\text{Cluster Weight}(T_j, T_k) = \frac{\sum_{i=1}^{n} d_{ijk}}{\sum_{i=1}^{n} d_{ij}} \times WeightingFactor(T_k) \quad (4)$$

$$\text{Weighting Factor}(T_k) = \frac{log\left(\frac{N}{df_{jk}}\right)}{\log N} \quad (5)$$

Hasil inilah yang akan dijadikan bobot pembentukan tesaurus sebuah kata. Semakin tinggi nilai *cluster weight* maka semakin mirip kata tersebut.

D. Algoritma Constraint Satisfaction Problem

Constraint Satisfaction Problem merupakan sebuah pendekatan untuk menyelesaikan suatu masalah dengan tujuan menemukan keadaan atau objek yang memenuhi sejumlah persyaratan atau kriteria. *Constraint Satisfaction Problem* adalah suatu permasalahan yang mencari nilai untuk set variabel (*finite*) yang memenuhi set *constraint*[5]. *Constraint Satisfaction Problem* memiliki komponen yaitu:

- Variabel
  Set *term co-occurrence*
- Domain
  Term yang memenuhi *constraint*
- *Constraint*
  a. *Node consistency*
  x_j konsisten jika dan hanya jika c_j sesuai pada *associate constraint network*.

$$c_j = \begin{cases} 1 \ if \ \sum_{i=0}^{n-1} w_{i,j} x_i \geq threshold \\ 0, if \ \sum_{i=0}^{n-1} w_{i,j} x_i < threshold \end{cases}$$

  b. *Associate constraint network*
  Associate constraint network sesuai jika dan hanya jika semua node dalam *associate constraint network* konsisten dan $\sum_j x_j < C$, dimana *C* adalah *threshold* dan ditentukan berdasarkan masukan. *C* adalah *threshold* yang digunakan untuk membatasi jumlah keluaran *term* tesaurus.

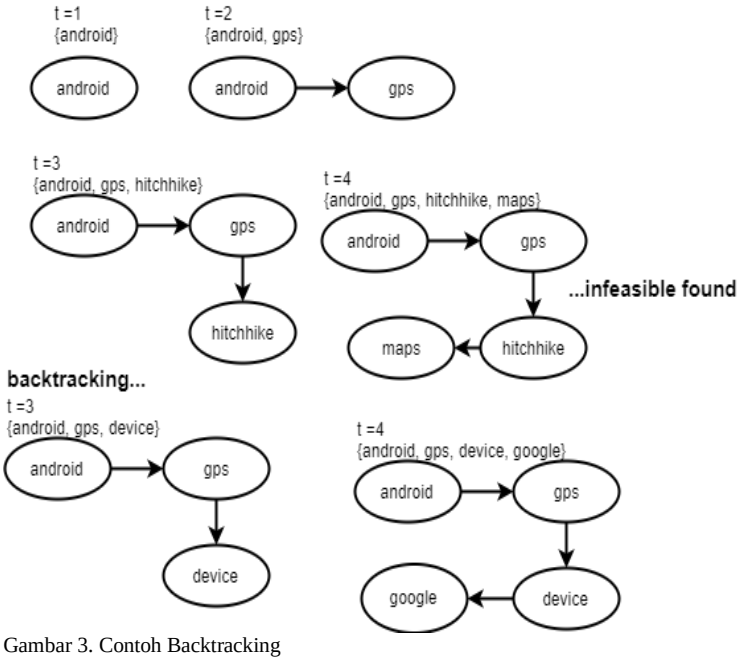
Solusi untuk *Constraint Satisfaction Problem* dapat ditemukan secara efisien dengan mencari secara sistematis melalui algoritma pencarian. Sebuah pendekatan alternatif adalah *backtracking*. *Backtracking* merupakan perbaikan dari algoritma *brute-force*, secara sistematis mencari solusi persoalan di antara semua kemungkinan solusi yang ada[6]. Metode *backtracking* hanya mempertimbangkan pencarian yang mengarah ke solusi saja. Sehingga waktu pencarian dapat dihemat.

Algoritma *backtracking*:

1. Inisialisasi. Nilai dari semua node di inisialisasikan sebagai 0.
2. Cari kemungkinan solusi.
   $$v_j = \sum_{i=0, i \neq j}^{n} W_{ij} x_i$$
   W_{ij} : *cluster weight* antara node *i* ke node
3. Menentukan jika solusi ditemukan
   Periksa *node consistency* pada semua node.
   IF semua *node consistent*
   Masuk pada himpunan solusi
   ELSE IF $\sum_j x_j < C$
   THEN solusi ditemukan
   ELSE // solusi *infeasible* ditemukan
   x(t) = 0 // *x(t)* penyebab *infeasible solution*
   Kembali ke proses 2 // *backtracking*
   ELSE // iterasi selanjutnya
   Kembali ke langkah 2.

Tabel 1.
Cluster Weight pada Pairing Term

w	Term1	Term2	Cluster Weight
$w_{1,2}$	android	gps	0,82
$w_{1,7}$	android	kendara	0,74
$w_{1,8}$	android	basis	0,67
$w_{2,3}$	gps	hitchihke	0,72
$w_{2,4}$	gps	device	0,71
$w_{2,7}$	gps	kendara	0,57
$w_{3,4}$	hitchihke	maps	0,56
$w_{3,6}$	hitchihke	google	0,42
$w_{3,1}$	hitchihke	android	0,39
$w_{5,6}$	device	google	0,75
$w_{5,9}$	device	tumpang	0,71
$w_{5,10}$	device	aplikasi	0,65



Gambar 3. Contoh Backtracking

Ilustrasi backtracking terdapat pada Gambar 3. Diketahui x_1 = android, x_2 = gps, x_3 = hitchhike, x_4 = maps, x_5= device, x_6 = google. x_1 akan menjadi *term* masukan dan dimasukan pada himpunan solusi. Nilai *cluster weight* pada pasangan kata dapat dilihat pada Tabel 1. Misal, nilai *threshold* untuk jumlah cluster weight = 2,25 dan *threshold C* (jumlah kata yang terbentuk) = 5. Pada Langkah 2, x_2 dimasukan ke himpunan solusi dan y_1 = 2 karena $w_{1,2}$ adalah yang terbesar. {android, gps} adalah solusi parsial. Solusi belum memenuhi *constraint* tetapi solusi saat ini adalah *feasible* solusi parsial. Algoritma kembali ke Langkah 2. Pada iterasi ini, x_3 adalah dimasukan dalam himpunan solusi, dan y_2 = 2 karena v_3 = $w_{1,2}$ + $w_{2,3}$ adalah yang terbesar. Solusi parsial sekarang menjadi {android, gps} *feasible* namun tidak memenuhi *constraint*. Solusi parsial dibeberapa iterasi berikutnya adalah {android, gps, hitchhike}, {android, gps, hitchhike, maps}, dan akhirnya

solusi *infeasible* ditemukan. Selanjutnya dilakukan *backtracking* mundur ke solusi parsial {android, gps, hitchhike}. Karena solusi parsial yang dihasilkan dari {android, gps, hitchhike} tidak layak, kita *backtrack* ke {android, gps} dan mendapatkan {android, gps, device} Dalam satu iterasi selanjutnya, solusi ditemukan sebagai {android, gps, device, google}. Algoritma *backtracking* terus mencari solusi sampai suatu *infeasible* solusi ditemukan. Kemudian menghapus node yang menyebabkan *infeasibile* dan melakukan *backtrack* solusi parsial sebelumnya.

III. Hasil dan Diskusi

Uji coba dilakukan dengan meminta pendapat 25 responden yang mahir dalam bahasa Indonesia dan bahasa Inggris, serta berada dalam lingkup bidang ilmu dari dokumen yang digunakan. Uji coba digunakan untuk melakukan perbandingan antara tesaurus yang terbentuk dengan CSP dan tidak menggunakan CSP. Rumus *precison* dapat dilihat pada (7). Hasil rata – rata *precision* CSP dan Non CSP dapat dilihat dalam Tabel 2.

Tabel 2.
Hasil Rata – Rata Precision CSP Dan Non CSP

	CSP (%)	Non CSP (%)
Precision	91,38	45,23

Uji coba juga dilakukan dengan melakukan pencarian dengan *query* ekspansi menggunakan tesaurus, maka dapat dihitung nilai *recall* pada (6), *precision* pada (7) dan akurasi pada (8).

$$Recall = \frac{TP}{TP + FN} \times 100\% \quad (6)$$

$$Precision = \frac{TP}{TP + FP} \times 100\% \quad (7)$$

$$Akurasi = \frac{TP + TN}{TP + FP + TN + FN} \times 100\% \quad (8)$$

Dimana TP adalah *True Positive* yang berarti data relevan yang diterima dari pencarian yang dicocokan dengan *ground truth*. TN adalah *True Negative* yang berarti data tidak relevan yang diterima dari pencarian yang dicocokan dengan *ground truth*. FN adalah *False Negativef* yang berarti jumlah data yang relevan pada *ground truth* yang tidak ada pada hasil pencarian. FP adalah *False Positive* yang berarti jumah data yang tidak relevan yang terdapat pada hasil pencarian. *Ground truth* adalah kunci jawaban hasil pencarian dokumen relevan yang dibuat secara manual. Hasil pengujian tersebut terdapat pada Tabel 3.

Tabel 3.
Hasil Pengujian

Recall(%)	*Precision*(%)	Akurasi(%)
86,67	100	86,67

IV. Kesimpulan

Dapat disimpulkan, bahwa tesaurus yang dibentuk *robust* untuk masukan *query*. Jumlah data mempengaruhi pembentukan tesaurus, karena apabila data semakin banyak, maka kata – kata yang terbentuk akan semakin bervariatif. Data yang memiliki topik serupa akan membentuk tesaurus yang lebih baik, karena tesaurus yang terbentuk berasal dari topik yang sama, maka tesaurus yang terbentuk akan semakin relevan. Tesaurus yang dibentuk dengan menggunakan CSP menghasilkan *precision* 91,38% sedangkan tesaurus yang dibentuk tanpa menggunakan CSP menghasilkan *precision* 45,23%. Pencarian menggunakan tesaurus menghasilkan *recall* 86,67% *precision* 100% dan akurasi 86,67%. Diharapkan untuk kedepannya algoritma pembentukan *parallel corpus* dilakukan dengan metode penterjemahan yang lebih baik lagi.

Daftar Pustaka

[1] D. W. Oard, "Alternative approaches for cross-language text retrieval," in *AAAI Symposium on Cross-Language Text and Speech Retrieval. American Association for Artificial Intelligence*, 1997, vol. 16.

[2] K. Wolk, "Noisy-parallel and comparable corpora filtering methodology for the extraction of bi-lingual equivalent data at sentence level," *ArXiv Prepr. ArXiv151004500*, 2015.

[3] N. Bel, C. H. Koster, and M. Villegas, "Cross-lingual text categorization," in *International Conference on Theory and Practice of Digital Libraries*, 2003, pp. 126–139.

[4] "HOWTO Fetch Internet Resources Using urllib2 — Python 2.7.13 documentation." [Online]. Available: https://docs.python.org/2/howto/urllib2.html. [Accessed: 07-Jul-2017].

[5] C. C. Yang, C.-P. Wei, and K. W. Li, "Cross-lingual thesaurus for multilingual knowledge management," *Decis. Support Syst.*, vol. 45, no. 3, pp. 596–605, 2008.

[6] C. C. Yang and K. W. Li, "An associate constraint network approach to extract multi-lingual information for crime analysis," *Decis. Support Syst.*, vol. 43, no. 4, pp. 1348–1361, 2007.