# **BAB II**

**KAJIAN PUSTAKA**

## **2.1 Kajian Hasil Penelitian Terdahulu**

Pada bagian ini akan dipaparkan mengenai kajian terdahulu yang disajikan sebagai bahan acuan dasar dalam penelitian. Selain kajian terdahulu, akan dijelaskan tentang kajian teori yang mendukung penelitian ini.

Penelitian ini membahas mengenai *Suffix Tree Clustering* dengan *WordNet* dan *Named Entity Recognition* untuk pengelompokan dokumen dilakukan oleh (Wijoyo *et al.*, 2017). Penelitian tersebut mengembangkan metode baru STC terhadap pengelompokan dokumen. Metode yang dikembangkan memiliki beberapa tahapan, yaitu preprocessing dokumen dengan mengekstraksi *named entity* serta melakukan deteksi sinonim berdasarkan *WordNet.* Tahap kedua adalah pembobotan *term* dengan tfidf dan neridf. Tahap ketiga adalah melakukan *clustering* dokumen dengan menggunakan *Suffix Tree Clustering*. Metode STC mampu membuktikan dapat meningkatkan kualitas *clustering* dokumen menjadi lebih baik.

Penelitian yang dilakukan oleh (Sutadi Heru, 2016) membahas mengenai metode *Suffix Tree Clustering* yang dilakukan terhadap pengelompokan dokumen *history* mesin pencari. Dalam penelitian tersebut menghasilkan pengelompokan dokumen untuk dilakukan klasifikasi data, sehingga dapat membuktikan bahwa unsur penting dalam pengolahan data terutama data – data mentah yang diinginkan untuk suatu bentuk pola – pola kumpulan data. Akan tetapi pada penelitian ini belum dapat menangani lebih dari satu Bahasa untuk pengelompokan hasil pencarian.

Penelitian yang dilakukan oleh (Tuwohingide *et al.*, 2016) juga membahas metode *Suffix Tree Clustering* terhadap pengelompokan dokumen web berbahasa indonesia. Dalam penelitian ini bertujuan untuk mengelompokan dokumen web berbahasa Indonesia berdasarkan *pharse* pada proses pemilihan *base cluster*. Metode yang digunakan pada penelitian ini mampu meningkatkan hasil pengelompokan dokumen yang semula menggunakan *scoring pharse* dengan kombinasi *document frequency* dan panjang *pharse* yang terlabeli pada STC tradisional. Metode tersebut juga berhasil mempertahan kan *overlapping cluster* yang merupakan kelebiha STC.

Hal tersebut juga dikuatkan dengan yang dilakukan oleh (Tri, Soedijono and Armadyah, 2015) bahwa metode STC dapat diterapkan pengelompokan abstrak skripsi. Hal tersebut dibuktikan dengan pengujian metode *Suffix Tree Clustering* dan *Singular Value Decomposite* yang dapat digunakan dalam abstrak menggunakan kata kunci menjadi tiga kelompok yaitu Komputer Akutansi, *Ebusines*, dan Sistem Informasi, hal tersebut mendapatkan nilai keakurasian sebesar 87,67%.

Setelah dipaparkan mengenai kajian terdahulu yang disajikan sebagai bahan acuan dalam penelitian maka menurut penelitian (Wijoyo *et al.*, 2017), (Sutadi Heru, 2016), (Tuwohingide *et al.*, 2016), (Tri, Soedijono and Armadyah, 2015) yang mampu memaparkan tentang metode *suffix tree clustering* (STC), memaparkan bahwa metode algoritma STC dapat digunakan dalam menentukan pola yang terbukti dapat membantu proses pengelompokan data serta dapat melakukan klasifikasi data sesuai dengan kategori atau jenis data yang digunakan, selain itu metode STC dapat menghasilkan nilai akurasi yang cukup tinggi dalam melakukan pengelompokan data yang sesuai dengan klasifikasi data tersebut. Dari kesimpulan penelitian metode STC, maka topik yang akan dikembangkan dalam penelitian ini menggunakan metode STC sebagai penerapan dalam mengelompokan judul penelitian dosen di ruang lingkup Universitas Ahmad Dahlan.

Tabel 2.1 Tabel review jurnal

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Penelitian** | **Peneitian 1** | **Penelitian 2** | **Penelitian 3** | **Penelitian 4** | **Yang diteliti** |
| Penulis | Satrio Hadi Wijoyo1, Admaja Dwi Herlambang2, Fahrur Rozi3, Septiyan Andika Isanta4 | Heru Sutadi | Desmin Tuwohingide1, Mika Parwita2, Agus Zainal Arifin3, Diana Purwitasari4 | Lina Tri Andaru1, Bambang Soedijono W2, Armadyah Amborowati3 | Nova Anggraini |
| Judul Naskah | Optimasi *Suffix Tree Clustering* dengan *Wordnet* dan *Named Entity Recognition* untuk Pengelompokan Dokumen | Implementasi *Suffix Tree Clustering* untuk Pengelompokan Dokumen yang telah di Akses Melalui Mesin Pencarian Google | Efisiensi *Phrase Suffix Tree* dengan *Single Pass Clustering* untuk Pengelompokan Dokumen Web Berbahasa Indonesia | Pengelompokan Abstrak Skripsi Menggunakan Metode *Suffix Tree Clustering* dan *Singular Value Decomposition* | Pengelompokan Judul Penelitian Dosen Menggunakan Metode *Suffix Tree Clustering* |
| Nama Jurnal | Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK) | Jurnal Ilmiah INFOTEK | JURNAL TEKNOLOGI TECHNOSCIENTIA | Seminar Nasional Teknologi Informasi dan Multimedia | - |
| ISSN | p : 2355-7699  e : 2528-6579 | 2502-6968  (Media Cetak) | 1979-8415 | 2302-3805 | - |
| Volume | 4 | 1 | 8 | - | - |
| Nomor | 4 | 1 | 2 | - | - |
| Tahun Penelitian | 2017 | 2016 | 2016 | 2015 | 2018 |
| Metode | *Suffix Tree Clustering* dengan *Wordnet* dan *Named Entity Recognition* | *Suffix Tree Clustering* | *Suffix Tree* dengan *Single Pass Clustering* | *Suffix Tree Clustering* dan *Singular Value Decomposition* | *Suffix Tree Clustering* |
| Kesimpulan | Penelitian ini menunjukkan bahwa metode yang diusulkan dapat melakukan pengelompokan dokumen dengan sangat baik. WordNet (synonym detction) dapat mendeteksi kata yang beda penulisan tapi sama makna. NER dapat mendeteksi dokumen yang mempunyai entitas. Selain itu, WordNet dan NER dapat digunakan untuk optimasi clustering dokumen menggunakan STC. | 1. Pengelompokan tree clustering terdiri dari beberapa prosedur diantara nya, Documen cleaning, steming data, identifikasi base cluster, dan pengkombinasian base cluster  2. Untuk menerapka metode STC ada beberapa langkah di antara nya dengan pencarian shared phrase untuk semua berita yang dikoleksi,dan mengkobinasikan dua base cluster kedalam satu cluster  3. Bahasa pemrograman yang di bangun dalam pengelompokan metode STC yaitu visual basic 2008. | Metode yang diusulkan menghasilkan nilai F-Measure yang lebih tinggi dibandingkan dengan STC tradisional. Efisiensi phrase pada proses pemilihan base cluster dengan kombinasi perhitungan tf dan df mampu meningkatkan hasil pengelompokan dokumen yang semula menggunakan scoring phrase dengan kombinasi df dan panjang phrase yang terlabeli pada STC tradisional. Metode yang diusulkan juga berhasil mempertahankan overlapping cluster yang merupakan kelebihan STC. | Akurasi yang sesuai diperoleh pada 100 data abstrak dengan menggunakan reduksi dimensi 30% yaitu akurasi 87.67% dengan hasil 54 data pada kelompok Sistem Informasi Perusahaan/ Instansi , 14 data terkelompok dalam Ebusines dan 26 data terkelompok dalam Komputer akutansi (pengujian recall adalah 0.33 dan presisisebesar 0.37). | Data yang digunakan dalam penelitian ini adalah data Penelitian Dosen UAD tahun 2015 sampai tahun 2017. Hasil dari penelitian ini berupa sebuah sistem yang dapat menampilkan hasil pengelompokan judul penelitian dosen berdasarkan bidang penelitiannya dan diuji dengan tingkat akurasi. |

## **2. 2 Landasan Teori**

### **2.2.1 Pengelompokan**

Pengelompokan data menjadi sejumlah kategori juga dapat dilakukan menggunakan metode *clustering*. Berbeda dengan metode klasifikasi, *clustering* mengelompokkan data hanya berdasarkan fitur-fitur yang ada pada data tersebut. Berdasarkan sifat tersebut, *clustering* tidak memerlukan data latih yang telah diketahui kelasnya. Oleh karena itu, proses pembelajaran pada *clustering* bersifat mandiri, yang sering disebut dengan istilah ***unsupervised learning.***

Tujuan utama dari proses *clustering* adalah membagi sekumpulan data menjadi sekumpulan grup (*cluster)* sehingga data-data dalam satu *cluster* memiliki banyak kemikiripan namun berbeda dengan data-data yang berada pada *cluster* lainnya. Kemiripan antar data dapat dihitung menggunakan berbagai metode pengukuran jarak.

Teknik *clustering* memiliki banyak kegunaan dalam kehidupan sehari-hari. Mesin pencarian informasi, seperti Google dan Yahoo, dapat mengelompokkan hasil pencarian sehingga pengguna dapat menelusuri hasil pencarian dari kelompok yang relevan saja. Hal ini dapat mengurangi waktu pencarian dan memperbesar peluang mendapatkan hasil yang sesuai. (Adinugroho and Sari, 2018).

### **2.2.2 Penelitian**

Penelitian adalah suatu kegiatan ilmiah dalam memecahkan masalah dengan cara sistematis yang telah ditetapkan untuk mencapai tujuan yang telah dirumuskan.

Sebagian orang menganggap bahwa metode penelitian terdiri dari berbagai teknik penelitian, dan sebagian lagi menyamakan metode penelitian dengan teknik penelitian. Tetapi yang jelas, metode atau teknik penelitian apa pun yang kita gunakan, baik kuantitif ataupun kualitatif, haruslah sesuai dengan kerangka teoretis yang kita asumsikan. (Muh. Fitrah and Dr. Luthfiyah, 2017).

### **2.2.3 Python**

(Enterprise, 2016) menyatakan bahwa Python merupakan bahasa pemrograman yang bersifat interpretative. Dibandingkan dengan Bahasa pemrograman lainnya, Python termasuk mudah dipelajari karena merupakan salah satu Bahasa high level programming. Python dapat diperoleh dan dipergunakan secara bebas oleh siapapun bahkan bagi developer yang menggunakan Bahasa pemrograman Python untuk kepentingan komersial.

Python merupakan pemrograman multiplatform. Itu artinya, Python dapat dijalankan di berbagai platform sistem operasi yaitu Linux / Unix, Windows, Mac OS X, Java Virtual Machine, OS /2, Amiga, Palm, Symbian (untuk produk-produk Nokia). Dengan begitu, baik pengguna Macintosh, Windows, maupun Linux dapat membuat berbagai jenis aplikasi menggunakan Python tanpa masalah.

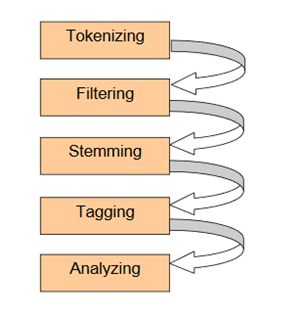
Python dikembangkan oleh Guido VanRossum (programmer kelahiran Belanda) pada tahun 1990 di CWI, Amsterdam sebagai kelanjutan dari bahasa pemrograman ABC. Nama Python dipilih oleh Guido sebagai nama bahasa ciptaannya karena kecintaan Guido pada acara televsi Monty Python’s Flying Circus. Saat ini perkembangan python terus dilakukan oleh sekelompok programmer yang dikoordinasi oleh Gudio dan Python Software Foundation. Sekarang, distribusi Python sudah mencapai versi 2.6.1 dan versi 3.0. beberapa fitur dan kelebihan yang dimiliki Python adalah :

1. Memiliki koleksi kepustakaan yang banyak. Itu artinya, telah tersedia modul-modul ‘siap pakai’ untuk berbagai keperluaan.
2. Memiliki struktur Bahasa yang jelas, sederhana, dan mudah dipelajari.
3. Berorientasi objek.
4. Memiliki sistem pengelolaan memori otomatis (garbage collection) seperti halnya Java.
5. Bersifat Modular sehingga mudah dikembangkan dengan menciptakan modul-modul baru, baik dibangun dengan Bahasa Python maupun C/C++.

### **2.2.4 Text Mining**

*Text Mining* secara umum adalah teori tentang pengolahan koleksi dokumen dalam jumlah besar yang ada dari waktu ke waktu dengan menggunakan beberapa analisis, tujuan pengolahan teks tersebut adalah mengetahui dan mengekstrak informasi yang berguna dari sumber data dengan identifikasi dan eksplorasi pola menarik dalam kasus *Text Mining*, sumber data yang dipergunakan adalah kumpulan atau koleksi dokumen tidak terstuktur dan memerlukan adanya pengelompokan untuk diketahui informasi sejenis. (Somantri and Wiyono, 2016).

*Text Mining* menurut Han & Kamber, adalah satu langkah dari analisis teks yang dilakukan secara otomatis oleh komputer untuk menggali informasi yang berkualitas dari suatu rangkaian teks yang terangkum dalam sebuah dokumen. Prosedur utama dalam metode ini terkait dengan menemukan kata-kata yang dapat mewakili isi dari dokumen untuk selanjutnya dilakukan analisis keterhubungan antar dokumen dengan menggunakan metode statistik tertentu seperti analisis kelompok, klasifikasi dan asosiasi. Tahapan dalam *Text Mining* secara umum diantaranya adalah *tokenizing, Filtering, Stemming, tagging, dan analyzing.* (Somantri and Wiyono, 2016).



**Gambar 2.1 : Tahapan *Text Mining***

Penjelasan Tahapan *Text Mining* :

1. ***Tokenizing***

Pada *tokenizing* terdapat beberapa proses yang harus dilakukan yaitu merubah semua huruf besar menjadi kecil (*text to lowercase*). Proses selanjutnya adalah penguraian, proses penguraian yang dimaksud adalah membagi teks menjadi kumpulan kata tanpa memperhatikan keterhubungan diantara kata satu dengan yang lain serta peran dan posisinya pada kalimat, karakter diterima dalam kumpulan kata menurut abjad. Sedangkan untuk perulangan kata dalam Bahasa Indonesia akan terbagi menjadi dua kata. (Wijaya and Santoso, 2016).

*Tokenizer* menerima input string dan memilahnya menjadi *token* (unit terkecil) sebagai penciri dokumen dengan aturan sebagai berikut:

1. *Token* dipisahkan oleh karakter whitespace (spasi)
2. Tanda baca(seperti ‟!‟, ‟?‟, ‟.‟, ‟,‟) dihilangkan
3. Suatu *token* dimulai dengan huruf atau angka

Output dari *Tokenisasi* adalah *token* serta informasi tambahan informasi lain seperti frekuensi kata, posisi kata dalam dokumen. (Lumbanraja, 2013).

1. ***Filtering***

*Filtering* adalah tahap mengambil kata-kata penting dari hasil tahap *token*izing. *Filtering* dapat dilakukan dengan menghilangkan *stoplist/stopword* (kata-kata yang tidak deskriptif, seperti kata “yang” dan “dari”). (Imbar *et al.*, 2014)

Filtering merupakan proses seleksi terhadap kata-kata yang dihasilkan dari proses tokenizing, dapat dilakukan dengan algoritma *stoplist* maupun *wordlist*. Algoritma *stoplist* akan membuang kata-kata yang tidak penting seperti kata ganti, kata keterangan, kata sambung, kata depan dan kata sandang. Sebaliknya, algoritma *wordlist* akan menyimpan kata-kata yang penting. (Prilianti and Wijaya, 2014).

Kata-kata yang dihapus (*stopwords*) adalah sebagai berikut : 'yang', 'untuk', 'pada', 'ke', 'para', 'namun', 'menurut', 'antara', 'dia', 'dua', 'ia', 'seperti', 'jika', 'jika', 'sehingga', 'kembali', 'dan', 'tidak', 'ini', 'karena', 'kepada', 'oleh', 'saat', 'harus', 'sementara', 'setelah', 'belum', 'kami', 'sekitar', 'bagi', 'serta', 'di', 'dari', 'telah', 'sebagai', 'masih', 'hal', 'ketika', 'adalah', 'itu', 'dalam', 'bisa', 'bahwa', 'atau', 'hanya', 'kita', 'dengan', 'akan', 'juga', 'ada', 'mereka', 'sudah', 'saya', 'terhadap', 'secara', 'agar', 'lain', 'anda', 'begitu', 'mengapa', 'kenapa', 'yaitu', 'yakni', 'daripada', 'itulah', 'lagi', 'maka', 'tentang', 'demi', 'dimana', 'kemana', 'pula', 'sambil', 'sebelum', 'sesudah', 'supaya', 'guna', 'kah', 'pun', 'sampai', 'sedangkan', 'selagi', 'sementara', 'tetapi', 'apakah', 'kecuali', 'sebab', 'selain', 'seolah', 'seraya', 'seterusnya', 'tanpa', 'agak', 'boleh', 'dapat', 'dsb', 'dst', 'dll', 'dahulu', 'dulunya', 'anu', 'demikian', 'tapi', 'ingin', 'juga', 'nggak', 'mari', 'nanti', 'melainkan', 'oh', 'ok', 'seharusnya', 'sebetulnya', 'setiap', 'setidaknya', 'sesuatu', 'pasti', 'saja', 'toh', 'ya', 'walau', 'tolong', 'tentu', 'amat', 'apalagi', 'bagaimanapun'.

1. ***Stemming***

*Stemming* merupakan proses penghilangan/pemotongan *prefiks* (awalan) dan *sufiks* (akhiran) dari kata dan istilah-istilah dokumen. *Stemming* diakukan atas dasar asusmi bahwa kata-kata yang memilik stem yang sama memiliki makna dasar yang sama. Teknik *Stemming* dapat dikategorikan menjadi 3, yaitu:

1. Berdasarkan aturan dalam bahasa tertentu
2. Berdasarkan kamus
3. Berdasarkan kemunculan bersama

Salah satu tujuan utama dilakukan proses *Stemming* adalah meningkatkan efesiensi. *Stemming* mengurangi jumlah kata-kata unik dalam indeks sehingga menghemat sumber daya komputasi dan sumber daya penyimpanan. (Lumbanraja, 2013).

Dalam dokumen Bahasa Indonesia proses stemming sangat diperlukan sebelum proses text mining karena Bahasa Indonesia memiliki *prefixes, suffixes, infexes* dan *confixes* yang membuat suatu kata dasar dapat berubah manjadi banyak bentuk dan akibatnya membuat pencarian kata dasar menjadi sulit. Berikut adalah arti dan contoh dari imbuhan dalam Bahasa Indonesia (Ngurah, Nata and Yudiastra, 2017):

1. *Sufiks* (Akhiran) adalah *afiks* yang ditambahkan pada bagian belakang kata dasar, misal “-an, kan,” dan “–i”;
2. *Prefiks* (Awalan) adalah imbuhan yang ditambahkan pada bagian awal sebuah kata dasar atau bentuk dasar; awalan: “per-” adalah yang paling produktif dalam bahasa Indonesia
3. *Konfiks* (*sufiks* dan *prefiks*) *afiks* tunggal yang terjadi dari dua unsur yang terpisah (misal “ke-...an” dalam kata “kemerdekaan”).

Salah satu library yang bisa digunakan dalam melakukan proses stemming bahasa indonesia adalah dengan menggunakan Library Python Sastrawi . Library ini merupakan pengembangan dari Library PHP Sastrawi dimana library tersebut menerapkan Algoritma Nazief dan Adriani . Tahapan algoritma tersebut meliputi :

1. Langkah pertama adalah memeriksa apakah kata tersebut merupakan akar kata (root) terdapat dalam daftar akar kata (root). Kita kata tersebut merupakan akar kata, maka proses dihentikan pada tahap pertama ini.
2. Menghilangkan *Inflection Suffixes* (“-lah”, “-kah”, “-ku”, “-mu”, atau “-nya”). Jika kata berupa particles (“-lah”, “-kah”, “-tah” atau “-pun”) maka langkah ini diulangi lagi untuk menghapus Possesive Pronouns (“-ku”, “-mu”, atau “-nya”).
3. Menghilangkan *derivational suffix* (imbuhan turunan). Hilangkan imbuhan -i, -kan, -an.
4. Menghilangkan *derivational prefix* (awalan turunan). Hilangkan awalan be-, di-, ke-, me-, pe-, se- dan te-.
5. Bila dari langkah 4 di atas belum ketemu juga. Maka lakukan analisis apakah kata tersebut masuk dalam tabel diambiguitas kolom terakhir atau tidak.
6. Bila dari langkah 4 di atas belum ketemu juga. Maka lakukan analisis apakah kata tersebut masuk dalam tabel diambiguitas kolom terakhir atau tidak.
7. Bila semua proses di atas gagal, maka algoritma mengembalikan kata aslinya.
8. ***Tagging***

Tahapan *tagging* adalah tahap mencari bentuk awal/root dari tiap kata lampau atau kata hasil *Stemming* yang bukan Bahasa Indonesia.

1. ***Analyzing***

Tahap *analyzing* merupakan tahap penentuan seberapa jauh keterhubungan antar kata-kata antar dokumen yang ada. Tahap ini menggunakan algoritma *term frequency* (tf), *invers document frequency* (idf) dan kombinasi perkalian antara keduanya (tf x idf).

Pembobotan kata dilakukan dengan cara menghitung *Term Frequency-Inverse Document Frequency* (tf-idf). Tf-idf adalah statistik numerik yang mengungkapkan bahwa seberapa penting sebuah kata dalam sebuah dokumen di koleksi. TF - IDF sering digunakan sebagai faktor pembobotan dalam pencarian informasi dan penambangan teks. Nilai tf-idf meningkat secara proporsional dengan berapa kali sebuah kata muncul dalam dokumen, namun menangkal frekuensi kata di dalam korpus. Ini bisa membantu mengendalikan fakta bahwa beberapa kata umumnya lebih umum daripada yang lain. (Vijayarani et al., 2016).

Dalam Poteet, Stephen R. dan Kao, A., menyatakan bahwa pembobotan kata dijabarkan oleh persamaan (2.1)

(2.1)

Dimana :

= bobot kata / term terhadap dokumen dj

= jumlah kemunculan kata / term ti dalam dokumen dj

N = Jumlah semua dokumen

N (ti) = Jumlah dokumen yang mengandung kata / term ti

Dalam(Intan, R. and Defeng, A, 2006). Robertson mengungkapkan bahwa tf.idf merupakan suatu cara untuk memberikan bobot hubungansuatu kata (*term*) terhadap dokumen. Metode ini menggabungkan dua konsep untuk perhitunganbobot yaitu, frekuensi kemunculan sebuah kata didalam sebuah dokumen tertentu dan *inverse*frekuensi dokumen yang mengandung kata tersebut. Frekuensi kemunculan kata didalamdokumen yang diberikan.Sehingga bobot hubungan antara sebuah kata dan sebuah dokumen akan tinggi apabila frekuensikata tersebut tinggi didalam dokumen dan frekuensi keseluruhan dokumen yang mengandung katatersebut yang rendah pada kumpulan dokumen (*database*).Rumus umum untuk Tf-Idf / *weight* (w) ditunjukkan oleh persamaan (2.2) dan (2.3)

(2.2)

(2.3)

Keterangan:

*wij* = bobot kata/*term tj* terhadap dokumen *di*

*tfij* = jumlah kemunculan kata/*term tj* dalam *di*

N = jumlah semua dokumen yang ada dalam *database*

n = jumlah dokumen yang mengandung kata/*term tj*

Berdasarkan rumus diatas, berapapun besarnya nilai *tfij*, apabila N = n maka akan didapatkan hasil 0 (nol) untuk perhitungan Idf. Untuk itu dapat ditambahkan nilai 1 pada sisi Idf, sehingga perhitungan bobotnya menjadi persamaan (2.4):

(2.4)

Keterangan:

*wij* = bobot kata/*term tj* terhadap dokumen *di*

*tfij* = jumlah kemunculan kata/*term tj* dalam *di*

N = jumlah semua dokumen yang ada dalam *database*

n = jumlah dokumen yang mengandung kata/*term tj*

### **2.2.5 Metode *Suffix Tree Clustering***

(Oren Zamir and Oren Etzioni, 1998)*, Suffix Tree Clustering* (STC) adalah pengelompokan waktu linier algoritma yang didasarkan pada identifikasi frasa yang ada umum untuk kelompok dokumen. Frasa dalam konteks kami adalah urutan-urutan satu atau lebih kata. Kami mendefinisikan *basis* *cluster* menjadi satu set dokumen yang berbagi frasa umum. STC memiliki tiga langkah logis:

* 1. *Document “Cleaning”*,
  2. *Identifying Base Clusters* menggunakan *Suffix Tree*, dan
  3. *Combining Base Clusters* ke dalam kelompok.

**Langkah 1 - *Document "Cleaning"***

Pada langkah ini, string teks yang mewakili masing-masing dokumen diubah menggunakan algoritma *stemming* (menghapus kata awalan dan akhiran dan mengurangi jamak ke tunggal). Batas kalimat (diidentifikasi melalui tanda baca dan HTML tag) adalah token yang ditandai dan bukan kata (seperti angka, Tag HTML dan sebagian besar tanda baca) dibuang. String dokumen asli disimpan, serta petunjuk dari mulai dari setiap kata dalam string yang diubah menjadi posisi dalam string asli. Ini memungkinkan untuk mengidentifikasi *frase* kunci dalam string yang diubah, untuk menampilkan teks asli untuk meningkatkan keterbacaan pengguna.

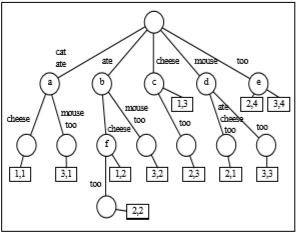
**Langkah 2 – *Identifying Base Clusters***

*Identifikasi cluster* dasar dapat dilihat sebagai pembuatan indeks frasa terbalik untuk dokumen yang dikoleksi. Ini dilakukan secara efisien menggunakan struktur data disebut *suffix tree* (Weiner, 73; Gusfield, 97). Struktur ini dapat dibangun dalam waktu linier dengan ukuran koleksi, dan dapat dibangun secara bertahap sebagai dokumen sedang dibaca (Ukkonen, 95). Ide menggunakan *Suffix Tree* untuk pengelompokan dokumen pertama kali diperkenalkan di (Zamir et. Al., 97). Di sini kami menyajikan peningkatan pengelompokan algoritma, yang memperkenalkan penggabungan klaster dasar (langkah tiga dari algoritma STC), dan membandingkannya menggunakan standar Metodologi IR untuk metode pengelompokan klasik di Web domain. *Suffix Tree* dari string *S* adalah *tree ringkas yang* berisi semua *suffix* dari *S.*  Dokumen sebagai string kata-kata, bukan karakter, sehingga sufiks mengandung satu atau lebih utuh kata-kata. Dalam istilah yang lebih tepat:

* + 1. *Suffix Tree* adalah pohon yang di-rooting dan digerakkan.
    2. Setiap simpul internal memiliki setidaknya 2 anak.
    3. Setiap sisi diberi label dengan sub-string non-kosong dari *S* (maka itu adalah *tree*). Label dari suatu simpul didefinisikan penggabungan tepi-label pada jalur dari root ke node itu.
    4. Tidak ada dua sisi dari node yang sama yang dapat memiliki edge-labels yang dimulai dengan kata yang sama (karena itu *ringkas*).
    5. Untuk setiap *suffix* *s* dari *S* , ada akhiran simpul yang label sama dengan *s* .

*Suffix Tree* dari kumpulan *string* adalah *tree* yang ringkas mengandung semua akhiran dari semua *string* dalam koleksi. Setiap akhiran-simpul ditandai untuk menunjuk dari *string* mana (atau *string*) itu berasal dari (*yaitu*, label *suffix* itu node adalah *suffix* dari *string* itu).

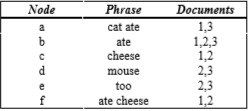
Gambar 2 adalah contoh dari *suffix tree* dari serangkaian *string* "*cat ate cheese* ", "*mouse ate cheese too*" dan "*cat ate mouse too*". Simpul dari *Suffix Tree* digambar sebagai lingkaran. Masing-masing *suffix* node memiliki satu atau lebih kotak yang melekat padanya *string* (s) itu berasal dari.Angka pertama di setiap kotak menunjuk string asal, angka kedua menandakan sufiks mana dari label *string* yang akhiran-simpul. Beberapa node pada gambar yang diberi label kata *f* untuk referensi lebih lanjut.



**Gambar 2:** *Suffix Tree* dari string "cat ate cheese", "mouse ate cheese too" dan "cat ate mouse too".

Setiap simpul dari *suffix tree* mewakili sekelompok dokumen dan frasa yang umum bagi mereka semua. Itu label simpul mewakili frasa umum, sekumpulan dari dokumen penandaan akhiran-node yang merupakan keturunan dari node membentuk grup dokumen. Karena itu, setiap node merupakan klaster basis. Selanjutnya, semua pangkalan mungkin kluster (berisi 2 dokumen atau lebih) muncul sebagai node di *suffix tree.* .Tabel 3 daftar enam node bertanda dari contoh ditunjukkan pada Gambar 2 dan basis terkait kelompok.

**Tabel 3:** 6 node dari contoh yang ditunjukkan pada Gambar 2 dan klaster basis terkait.



Setiap kelompok dasar diberi skor yang merupakan fungsi dari jumlah dokumen yang dikandungnya, dan kata-kata itu membuat ungkapannya. Skor *s (B)* dari base cluster *B* dengan frase *P* diberikan oleh:



dimana *| B |* adalah jumlah dokumen dalam kelompok basis *B*, dan *| P |* adalah jumlah kata dalam *P* yang memiliki skor bukan nol (*yaitu*, panjang kalimat yang efektif). *stoplist* yang dilengkapi dengan kata-kata khusus Internet (misalnya, "sebelumnya", "java", "frame" dan "mail"). Kata-kata muncul di *stoplist*, atau yang muncul terlalu sedikit (3 atau kurang) atau terlalu banyak (lebih dari 40% dari koleksi) dokumen menerima skor nol. Fungsi *f* menghukum kata tunggal frasa, linear untuk frasa yang panjangnya dua hingga enam kata, dan menjadi konstan untuk frasa yang lebih panjang.

**Langkah 3 – Combining Base Clusters**

Dokumen dapat berbagi lebih dari satu frasa. Akibatnya, itu kumpulan dokumen dari kluster basis yang berbeda dapat tumpang tindih dan mungkin bahkan identik. Untuk menghindari proliferasi hampir cluster identik, langkah ketiga dari algoritma menggabungkan basis klaster dengan tumpang tindih yang tinggi dalam kumpulan dokumen mereka (frasa tidak dipertimbangkan dalam langkah ini). Pada Gambar 1, misalnya, cluster atas dihasilkan dari penggabungan dua cluster dasar berlabel "Puerto Rico" dan "Musik Latin" berdasarkan pada merekakumpulan dokumen tumpang tindih.

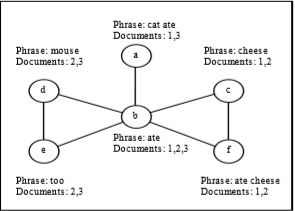
Mendefinisikan ukuran kemiripan biner antara basis klaster berdasarkan tumpang tindih set dokumen mereka. Diberikan dua kelompok basis *B m* dan *B n* , dengan ukuran *| B m |* dan *| B n |* masing-masing, dan | *B m* ∩ *B n* | mewakili jumlah dokumen umum untuk kedua kelompok dasar, mendefinisikan kesamaan *B m* dan *B n* menjadi 1 jika:

• *| B m* ∩ *B n | / | B m |* > 0,5 dan

• *| B m* ∩ *B n | / | B n |* > 0,5

Jika tidak, kesamaan mereka didefinisikan sebagai 0.

Selanjutnya, kita melihat *grafik cluster* dasar, di mana node berada *cluster* dasar, dan dua node terhubung jika dua basis *cluster* memiliki kesamaan 1. Sebuah cluster didefinisikan sebagai suatu komponen yang terhubung dalam *grafik cluster* dasar. Setiap clusterberisi penyatuan dokumen dari semua klaster dasarnya. Gambar 4 mengilustrasikan grafik klaster dasar dari enam basis cluster dalam Tabel 3. Ada satu cluster dalam contoh ini.



**Gambar 4:** Grafik klaster dasar dari contoh yang diberikan pada Gambar 2 dan Tabel 3. Dalam contoh ini ada satu komponen terhubung, karena itu satu cluster. Perhatikan itu jika kata *makan* telah ada di stoplist kami, kelompok basis *b* akan dibuang karena itu akan memiliki skor 0, dan kemudian kami akan memiliki tiga terhubung komponen dalam grafik, mewakili tiga kluster.

Membiarkan dokumen muncul lebih banyak dari satu kluster mengakui bahwa dokumen itu rumit objek yang dapat dikelompokkan menjadi beberapa berpotensi tumpang tindih, tetapi secara internal koheren, kelompok. Ini sebenarnya alasannya banyak sistem IR menggunakan beberapa bentuk dot-produk mendokumentasikan ukuran kesamaan (sebagai lawan Euclidean jarak, misalnya): memungkinkan dokumen serupaMbeberapa dokumen atau centroid yang berbeda yang pada gilirannya bias sangat berbeda satu sama lain.

Di STC, karena dokumen dapat berbagi lebih dari satu frasa dengan dokumen lain, setiap dokumen mungkin muncul dalam jumlah kelompok basis. Oleh karena itu, dokumen dapat muncul di lebih dari satu cluster. Perhatikan bahwa tumpang tindih antar kluster tidak bisa terlalu tinggi, jika tidak mereka akan bergabung menjadi satu kelompok.