Jurnal Linguistik Komputasional
Indonesian Association for Computational Linguistics
JLK
Vol 6, No 1
Maret 2023

Penggunaan Pemodelan Topik dalam Sistem Temu Kembali Dokumen Termirip

Lucia D. Krisnawati[#1], Joseph F. Lim[*2], Gloria Virginia[#3]

[123]*Informatika, Universitas Kristen Duta Wacana*
Jl. Dr. Wahidin Sudiro Husodo No. 5-25, Yogyakarta
[1]krisna@staff.ukdw.ac.id
[2]josepph.fernando@ti.ukdw.ac.id
[3]virginia@staff.ukdw.ac.id

Abstrak— **Sistem temu kembali didesain untuk menemukan informasi yang relevan dengan kueri pengguna, sedangkan pencarian dan penemuan dokumen termirip secara leksikal maupun sintaksi masuk ranah sistem deteksi plagiasi dan Daur Ulang teks *(text reuse*) yang membutuhkan sistem temu kembali sebagai salah satu modul di awalnya. Perbedaan keduanya terletak di bentuk kueri, dimana sistem temu kembali menerima kueri dengan jumlah token yang terbatas, sedangkan kueri dalam sistem deteksi plagiasi diolah dari sebuah dokumen input. Penelitian ini mencoba membangun sistem temu kembali untuk menemukan kandidat dokumen termirip yang diperlukan oleh kedua sistem tersebut. Untuk itu, pembentukan kueri dokumen input dihasilkan dengan memanfaatkan sistem Pemodelan Topik Lexikat. Elemen dari kumpulan topik ini kemudian diindek dalam Inverted Index maupun menjadi kueri dari dokumen uji. Metrik kemiripan *Cosine* digunakan untuk mengukur kemiripan antara kueri dokumen uji dengan dokumen sumber yang telah diindeks. Evaluasi sistem dilakukan dengan menggunakan metrik Macro-Averaged F1 (MAF) dan Break-Even Point (BEP). Eksperimen dilakukan dengan menggunakan 1-50 topik di tiap dokumen uji yang berjumlah 474. Hasil eksperimen dengan 11 skenario pengujian menunjukkan nilai MAF tertinggi mencapai 0.32 – 0.33 saat menggunakan 1 topik sebagai kueri. Nilai ini relatif kecil karena tidak diterapkan nilai ambang dari persamaan *Cosine* sebagai proses penyaringan (*filtering*) dokumen yang akan dievaluasi. Jumlah dokumen minimal yang diperlukan untuk mencapai nilai BEP tertinggi adalah 243. Jumlah dokumen ini bisa dijadikan rekomendasi sebagai nilai ambang dalam proses penyaringan dokumen.**

Kata kunci— **Pemodelan Topik, Sistem Temu Kembali, Lexikat, Kemiripan Dokumen, pembentukan kueri.**

I. PENDAHULUAN

Istilah Temu Kembali Dokumen yang merupakan alih bahasa dari *Information Retrieval* (IR) memberi kesan pada kita bahwa terminologi ini lebih menekankan pada dokumen sebagai objek daripada informasi. Baeza-Yates dan Ribeiro-Neto [1] mendefinisikan IR sebagai "area riset yang terkait dengan representasi, penyimpanan, organisasi, dan akses ke informasi seperti dokumen, situs Web, katalog daring, berkas terstruktur dan semi-terstruktur, serta objek multimedia", sedangkan Manning, Raghavan, dan Schütze [2]lebih menekan pada proses penemuan "material (biasanya dokumen) tak-terstruktur yang memenuhi kebutuhan informasi dari korpus dengan ukuran yang sangat besar (yang biasanya disimpan di komputer)".

Ada dua (2) hal yang menjadi kata kunci dalam kedua definisi IR tersebut. Yang pertama adalah informasi yang terkandung di dalam dokumen dan yang kedua adalah kebutuhan informasi. Pemahaman dokumen sebagai wadah informasi di sini tidak dibatasi pada berkas tekstual namun lebih ke artefak baik secara fisik ataupun digital yang mengandung informasi dan yang diciptakan untuk kita baca atau konsumsi [3]. Sedangkan kata 'kebutuhan informasi' (*information need*) yang sering dijadikan dasar evaluasi dari sistem IR bisa saja disalah-interpretasikan bahwa setiap pengguna memiliki kebutuhan informasi yang jelas dan spesifik saat melakukan proses temu kembali. Faktanya tidaklah demikian. Betapa sering pengguna memiliki kesulitan dalam memformulasikan kueri sebagai wujud kebutuhan informasinya sehingga proses temu kembalipun tidak bisa memberikan hasil yang maksimal.

Permasalahan berikutnya terletak pada pemahaman 'kemiripan' antara dokumen (informasi) dengan kueri (kebutuhan informasi pengguna). Spektrum kemiripan atau jarak ini cukup luas cakupannya yakni dari yang hanya memiliki keterkaitan topik (*topically related*), kemiripan di tingkat sintak sampai ke kemiripan semantik. Krisnawati dan Schulz [4] mencatat bahwa tugas sistem IR adalah menemukan dokumen yang topiknya terkait dengan kueri di tataran leksikal dengan panjang terbatas, sedangkan penemuan dokumen termirip dengan kueri dokumen masuk ranah sistem deteksi duplikat dan hampir duplikat (*near-duplicate*). Kemiripan dokumen dengan kueri dalam bentuk dokumen dalam sistem Deteksi duplikat atau hampir duplikat berada di tataran sintaksis. Spektrum kemiripan yang lebih luas, dari tataran sintaksis sampai dengan semantik masuk ranah sistem Deteksi Daur Ulang Teks (text reuse) dan sistem Deteksi Plagiasi [4]. Dalam konteks ini, jenis dan panjang kueri serta spektrum tataran

Submitted 24-02-2022; Revised 27-02-2023; Accepted 12-03-2023

kemiripan menentukan bidang area penelitian dari sistem yang hendak dibangun.

Dalam dekade terakhir ini, muncullah area penelitian yang mencoba menemukan topik dari sebuah dokumen input secara otomatis. Ide sistem Pemodelan Topik ini didasarkan pada cara kita menulis sebuah dokumen, dimana kita memikirkan sebuah topik atau pokok pikiran kemudian memilih dan menyusun kata demi kata bagi topik tersebut [5]. Kitapun memiliki beberapa pokok pikiran atau topik saat menulis sebuah teks. Sistem Pemodelan Topik ini mencoba mendekonstruksi proses penulisan sebuah teks tersebut. Dengan demikian saat diberi input sebuah dokumen teks, sistem Pemodelan Topik akan memberikan luaran berupa kumpulan topik dimana tiap topiknya terdiri dari beberapa token yang memiliki keterkaitan leksikal dan semantik.

Penelitian ini mencoba menemukan dokumen yang memiliki kemiripan di tataran leksikal (topik) dan sintaksis dengan dokumen kueri. Kami berasumsi bahwa dengan menggunakan kumpulan topik yang digenerasikan dari sistem Pemodelan Topik sebagai kueri, maka dokumen yang memiliki keterkaitan erat di tataran leksikal maupun sintaksis bisa ditemukan. Penelitian ini bersinggungan erat dengan IR, perbedaannya terletak di panjang kueri yang diambil dari dokumen input serta pemrosesannya yang merupakan sebuah luaran dari sistem Pemodelan Topik. Sistem Pemodelan topik yang digunakan adalah Lexikat yang merupakan produk sebuah Start-up perusahaan di Singapore dimana peneliti memiliki hak akses penuh.

II. TINJAUAN PUSTAKA

Sistem Temu Kembali dokumen yang berusaha menemukan kemiripan di tingkat leksikal, biasanya dilakukan untuk menemukan dokumen yang memiliki keterkaitan topik seperti untuk penemuan pasal dan ayat dalam Kitab Undang-undang Hukum Pidana- KUHP [6], ataupun ayat-ayat Alkitab [7]. Karakteristik utama dari Sistem temu kembali ini terletak pada panjang kueri yang dimasukkan oleh pengguna secara langsung yang rata-ratanya kurang dari 10 token.

Sedangkan pencarian dokumen dengan kemiripan di tataran sintaksis (kalimat) bisa diterapkan untuk sistem detaksi plagiasi [8] atau verifikasi penulis [9]. Pencarian dokumen dengan kueri di tataran leksikal yang digenerasikan dari sebuah dokumen input telah terbukti mampu menemukan dokumen yang tidak hanya memiliki keterkaitan topik, namun juga kemiripan di tataran sintaksis dalam sistem deteksi plagiasi dan daur ulang teks [10].

Luaran Sistem Pemodelan Topik (SPT) sering digunakan sebagai masukan (*input*) bagi aplikasi lainnya. Beberapa penelitian berikut ini memanfaatkan luaran SPT untuk klasifikasi konten teks [5], Analisis sentimen [11], maupun untuk klasifikasi subjek email [12]. Sedangkan Anugerah dan Rosyid [13] memanfaatkan luaran SPT untuk temu kembali dokumen yang relevan dengan kueri untuk fitur pencarian di Portal Multimedia.

Perbedaan penelitian kami dibanding dengan penelitian yang menggunakan sistem Pemodelan Topik sebelumnya [13] terletak pada pemrosesan topik, penggunaannya dan model yang digunakan. Dalam [13], topik digenerasikan dari dokumen yang berisi tentang diskripsi video. Model yang digunakan untuk generasi topik adalah *Probabilistic Latent Semantic Analysis* (PLSA). Token dari topik ini kemudian dijadikan fitur dokumen tersebut dan dicocokkan dengan kueri pengguna. Berbeda dari [13], Haryanto dkk [14] berusaha menemukan dokumen sumber rujukan dari daur ulang teks (*text reuse*) dengan model yang lebih menekankan pada generasi kueri dari dokumen terdaur ulang. Ada dua tahap yang digunakan untuk mengekstraksi kueri dari dokumen uji, yakni segmentasi per 250 token, setelah itu dilakukan pembobotan token dengan *local term weighting*. Setiap segmen memiliki nilai ambang yang berbeda tergantung dari bobot lokal kata dalam segmen tersebut. Setelah itu, formulasi kueri dilakukan dengan mengabungkan token yang telah tersaring tadi, namun penyaringan tahap berikutnya dilakukan untuk mendapat-kan token unik bagi sebuah kueri [14].

Berdasarkan pemrosesan kueri, penelitian ini lebih mendekati penelitian [14], hanya kueri digenerasikan dari Lexikat dalam bentuk topik. Berbeda dari [13] yang menggunakan topik sebagai fitur dokumen sumber yang pendek, maka dalam penelitian kami, topik berfungsi sebagai kueri yang dihasilkan dari dokumen uji sekaligus fitur dokumen sumber. Tujuan utamanya adalah menemukan dokumen termirip di tataran leksikal maupun siktaksis yang memberi ruang adanya parafrase atau modifikasi lainnya. Untuk itu penelitian ini melakukan eksperimen dengan panjang kueri yang berbeda yakni dari dari jumlah topik yang berbeda dari dokumen uji.

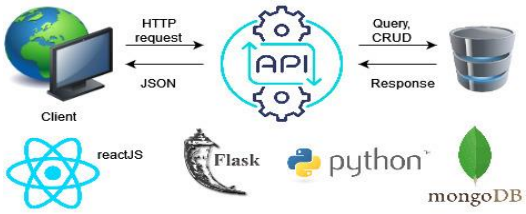
III. METODE PENELITIAN

A. Arsitektur Sistem Full Stack

Sistem Temu Kembali dokumen yang diusulkan ini berbasis Web sehingga perancangannya perlu memenuhi standard sistem pengembangan tumpukan penuh (*full stack development*). Tampilan antarmuka (*front-end*) sistem ini dibuat dengan framework React.JS. Sedangkan Aplication Programing Interface (API) dibangun dengan framework Flask dan Python untuk *API endpoint*.

Dari perspektif kasus penggunaan, maka saat pengguna memberi masukan dokumen sebagai kueri, maka kueri ini diubah menjadi sebuah HTTP request ke *API endpoint* yang berkomunikasi dengan sistem temu kembali. Sistem temu kembali ini dibangun dengan bahasa pemrograman Python dan menggunakan basis data MongoDB. Sistem temu kembali ini akan memproses kueri dokumen dan membandingkan dengan vektor fitur tiap dokumen sumber yang ada di basis data. Hasil pemrosesan ini dikembalikan sebagai respon berformat JSON yang kemudian ditampilkan kembali ke pengguna dalam bentuk yang

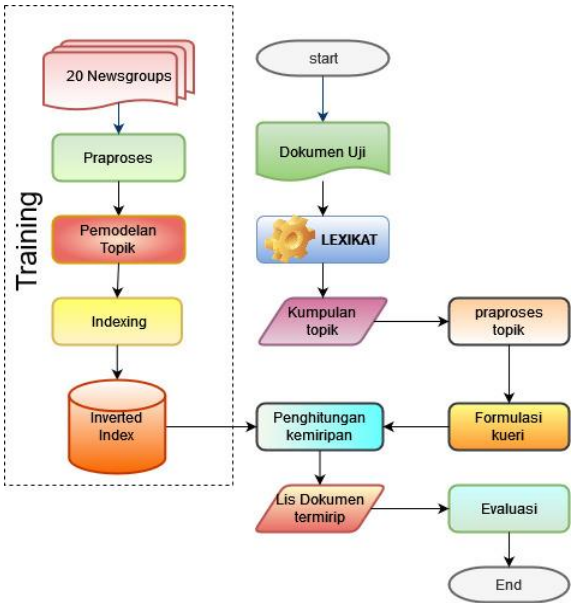
mudah terbaca. Gambar 1 menampilkan komunikasi antara antarmuka, API endpoint yang mengarah ke sistem Temu Kembali, dan basis data MongoDB.



Gambar 1 Arsitektur sistem dalam kerangka sisem *Full Stack*.

B. *Aristetur Sistem Temu Kembali*

Proses pembangunan sistem Temu Kembali yang mengolah sebuah dokumen sebagai kueri melalui sistem Pemodelan Topik dapat dipilah menjadi dua (2) bagian utama, yakni pembangunan *invereted index* yang bisa dianalogikan sebagai proses pelatihan, serta sistem temu kembali yang menerima kueri berupa topik yang disarikan dari sebuah dokumen uji. Dua proses utama ini ditampilkan di Gambar 2.



Gambar 2 Tahapan pembangunan sistem Temu Kembali yang memanfaatkan sistem Pemodelan Topik Lexikat.

Gambar 2 memperlihatkan kumpulan dokumen atau korpus yang diambil dari *20 Newsgroups* diproses dengan melakukan normalisasi konten tiap dokumen, kemudian dilakukan Pemodelan Topik menggunakan algoritma Lexikat yang dimodifikasi dan dilakukan *indexing* bagi topik beserta term dari tiap dokumennya yang merupakan proses standard pembentukan sistem Temu kembali. Luaran proses ini disimpan sebagai *inverted index* dalam basis data MongoDB.

Sistem kerja dari sistem temu kembali ini akan dimulai saat sebuah dokumen diinputkan ke sistem, maka dokumen uji tersebut menjadi input di sistem Pemodelan Topik Lexikat yang akan memberi luaran kumpulan topik yang terdiri dari beberapa term atau token. Maka praproses terhadap topik dilakukan untuk membentuk sebuah kueri yang kemudian akan dihitung nilai kemiripannya dengan tiap vektor fitur tiap dokumen yang ada di inverted index. Maka berdasarkan nilai kemiripan ini dibuatlah rangking dokumem yang ditemukan yang kemudian dilakukan evaluasi untuk melihat seberapa bagus kinerja sistem ini. Tiap tahapan proses pembangunan sistem ini akan diulas lebih lanjut di sub-bahasan selanjutnya.

C. *Pengumpulan data*

Penelitian ini memilih menggunakan data yang sudah tersedia dan bersifat publik daripada membangun sendiri. Alasan utamanya adalah himpunan data (dataset) tersebut sudah terlabeli dengan baik, teruji dan juga menghemat waktu penelitian. Data yang digunakan merupakan korpus yang sering digunakan sebagai eksperimen berbagai aplikasi teks dari Pembelajaran mesin (Machine Learning) seperti klasifikasi dan kategori teks. Korpus ini dikenal dengan nama 20 Newsgroups yang terdiri dari 18.846 dokumen yang masing-masing sampel memiliki label berupa sebuah topik dari dokumen tersebut. Data 20 Newsgroup ini bisa diakses dari laman www.kaggle.com atau bisa diimpor langsung dari Scikit-learn.

Menurut laman resmi korpus 20 newsgroups [1], data tersebut dikelompokkan dalam 20 kategori berdasarkan topiknya. Namun ada beberapa kategori yang memiliki keterkaitan topik yang sangat erat, dan ada yang sama sekali tidak memiliki keterkaitan topik. Sehingga dari 20 topik tadi dipartisi menjadi 6. Tabel 1 menunjukkan 6 area topic dari koprus 20 Newsgroups

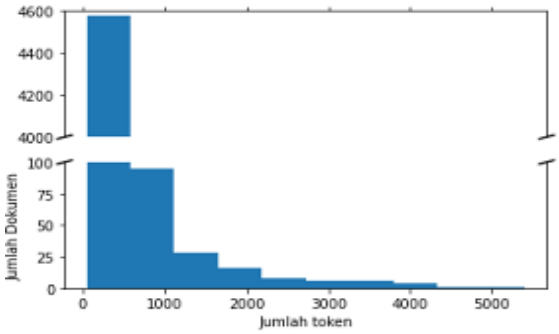
TABEL 1 ENAM (6) PARTISI DARI 20 KATEGORI TOPIK YANG ADA DI KORPUS 20 NEWSGROUPS

comp.graphics comp.os.ms-windows.misc comp.sys.ibm.pc.hardware comp.sys.mac.hardware comp.windows.x	rec.autos rec.motorcycles rec.sport.baseball rec.sport.hockey	sci.crypt sci.electronics sci.med sci.space
misc.forsale	talk.politics.misc talk.politics.guns talk.politics.mid-east	talk.religion.misc alt.atheism soc.religion.christian

[1] Alamat laman resmi 20 newsgroups adalah: http://qwone.com/~jason/20Newsgroups/

Berhubung dokumen data ini diambilkan dari berita, maka sebagain besar dokumen merupakan teks pendek dengan panjang dibawah 150 token. Untuk menyesuaikan dengan keperluan algoritma dari Lexikat, maka tidak semua dokumen dapat digunakan, hanya dokumen yang memiliki jumlah kata lebih dari 150. Berdasarkan kategori ini, maka hanya 19 kategori yang bisa digunakan baik sebagai dokumen sumber maupun dokumen uji yang jumlah totalnya mencapai 4.740.

Masih di seputar jumlah dokumen, dari 4.740 dokumen yang dijadikan penelitian ini, ada 4.500 dokumen yang memiliki jumlah token dibawah 500. Ini menandakan bahwa mayoritas dokumen yang digunakan untuk keperluan penelitian ini cukup pendek. Distribusi panjang dokumen yang digunakan dalam penelitian ini ditunjukkan di Gambar 3.



Gambar 3 Distribusi jumlah token dari korpus 20 Newsgroups yang digunakan dalam penelitian ini.

D. Pembentukan Inverted Index

Dalam sistem temu kembali ini, pembangunan Inverted Index bisa dianalogikan sebagai proses pelatihan untuk mendapatkan nilai fitur vektor bagi dokumen sumber atau latih. Pembangunan *Inverted index* yang terdiri dari token, kolokasi beserta nilai vektor yang tercipta dari topik diawali dengan prapemrosesan teks korpus 20 Newsgroups. Langkah-langkahnya adalah sebagai berikut:

1) *Prapemrosesan.*

Langkah awal dalam prapemrosesan teks di korpus 20 Newsgroups adalah pengubahan **semua token menjadi huruf kecil**. Berhubung teks di korpus 20 Newsgroups memiliki genre yang beragam termasuk berita dan email, maka langkah selanjutnya adalah **menghilangkan format email** dengan jawaban atau pesan yang diteruskan, namun isi teksnya dipertahankan. Jika dalam dokumen ditemukan **URL, tag HTML dan karakter spesial,** maka string yang masuk kategori tersebut akan dihilangkan dengan menggunakan pustaka Reguler Expression (re) di Python.

2) Pemodelan Topik

Untuk mendapatkan topik, kami menggunakan sistem Pemodelan Topik Lexikat yang merupakan produk dari sebuah perusahaan start-up di Singapore. Pemodelan topik pada Lexikat dibuat secara interaktif agar user dapat berinteraksi dengan topik yang dihasilkan. Algoritma yang digunakan oleh Lexikat tidak bisa kami jabarkan secara rinci di sini dengan alasan pelanggaran kode etik saat kami mendapatkan hak akses penuh ke kode program. Namun untuk beberapa fase yang dimodifikasi akan dipaparkan di sini.

Setelah teks dianggap bersih maka teks diinputkan ke Lexikat yang akan melakukan proses **tokenisasi**. Proses ini memakan waktu yang lama, karena bukan hanya token yang dihasilkan namun **2-5 gram** kata yang langsung dibentuk oleh algoritma Lexikat. 1-5 gram kata yang terbentuk ini kemudian dilatih berdasarkan korpus lexikat yang terdiri dari jutaan dokumen dari Wikipedia berbahasa Inggris. Pemrosesan ini **menghasilkan token dan kolokasi** atau frasa yang mayoritasnya terdiri dari 2-3 kata. Kolokasi ini kemudian diimpor dari Lexikat dengan menggunakan pustaka eksternal Python Pickle.

Penghapusan Stopwords dilakukan setelah proses pembentukan kolokasi karena dalam beberapa kolokasi dibutuhkan stopwords, seperti dalam "self defence", "an awful lot", atau "bless you". Luaran dari penghapusan stopword ini ditampung dalam variabel out_words_all_dirty yang menjadi dasar penghitungan term frequency (tf). Variabel ini masih mengandung kolokasi yang berisi stopword semua sebagai contohnya "it is", "there are", dll. Selain itu, kami menjumpai bahwa token yang membentuk kolokasi, tidak lagi ada di list out_words_all_dirty karena algoritma Lexikat akan mengabungkan token pembentuk kolokasi dan menghapus token tersebut di lis.

Normalisasi elemen out_words_all_dirty dilakukan dengan penghapusan kolokasi yang terbentuk dari stopword semua seperti "it is". Kemudian, modifikasi kode program dilakukan untuk mempertahankan pasangan token pembentuk kolokasi. Luaran yang dianggap daftar token dan kolokasi bersih beserta nilai frekuensinya ditampung dalam list of tuple out_words_all_clean. List inilah yang dijadikan dasar untuk pembentukan topik dalam proses.

Klasterisasi token dilakukan pada elemen out_words_all_clean untuk mendapatkan topik. Sebelum klasterisasi dijalankan, Lexikat menggunakan Dynamic_dict yang merupakan lexikon token dengan kata yang memiliki keterkaitan semantik hasil pelatihan dari korpusnya Lexikat. Gambar 4 menunjukkan salah satu term dengan kata yang berelasi semantik dalam Dynamic_dict-nya Lexikat.

```
expert
['expert', 'an_expert', 'specialist', 'researcher',
    'An_expert', 'technologist', 'criminologist', 'leading_expert',
    'analyst', 'legal_expert', 'psychologist', 'toxicologist', 'scientist',
    'consultant', 'epidemiologist', 'theorist']
```

Gambar 4 Salah satu contoh sebuah term dengan daftar token dan kolokasi yang memiliki relasi semantik dalam Dynamic_dict.

Klasterisasi token dan kolokasi berdasarkan leksikon Dynamic_dict ini akan menghasilkan kelompok token

beserta nilai tf. Setelah diamati, maka ada beberapa kelompok yang memiliki kesamaan. Contoh kelompok yang sebenarnya memiliki elemen yang sama ditunjukkan di Gambar 5. Tindakan yang dilakukan adalah melebur kelompok tersebut menjadi satu dan dipilihlah sebuah token atau kolokasi yang memiliki nilali tf tertinggi sebagai kategori atau topik. Dalam Gambar 5 nilai tf kelompok tersebut sama, maka tidak menjadi masalah token mana yang menjadi kategori. Contoh kelompok kata dengan kategori yang memiliki nilai tf tertnggi ditampilkan di Gambar 6 dengan kata *applications* yang menjadi kategori.

(‘knowledge’, {‘knowledge’:1, ‘understanding’:1})
(‘understanding, {‘understanding’:1, ‘knowledge’:1})

Gambar 5 Contoh Kelompok kata yang dapat digabungkan

(‘applications‘, {‘machines’:1, ‘technologies’:1, ‘applications’:4, ‘systems’:1, ‘tools’:3, ‘solution’:1, ‘application’:2})

Gambar 6 Contoh kelompok kata dengan kategori ‘applications’

Setelah menyatukan kelompok yang sama dan menghapus kelompok kata yang sudah disatukan, langkah berikutnya adalah mengurutkan kelompok kata berdasarkan total frekuensinya dan kata tersebut dijadikan topik. Gambar 7 menunjukkan 4 topik di urutan teratas yang dihasilkan oleh Lexikat. Yang perlu diperhatikan adalah topik yang dihasilkan oleh Lexikat merupakan hasil pemrosesan dari sebuah dokumen tunggal.

```
chatbot,,,,,,
applications,tools,application,machines,technologies,systems,solution
natural_language_processing,natural_language,semantic,machine_learning,,,
tasks,process,task,action,step,,
```

Gambar 7 Contoh 4 topik diperingkat teratas dari sebuah dokumen.

3) Indexing

Topik yang dihasilkan oleh Lexikat dari tiap dokumen dari korpus 20 Newsgroup tidak serta merta dijadikan fitur dokumen. Ini disebabkan Lexikat masih menggunakan pembobotan lokal yang bergantung pada tf di satu dokumen tertentu. Agar topik bisa menjadi fitur yang merepresentasikan dokumen, maka dibutuhkan pembobotan global yang menghitung bobot sebuah token/kolokasi yang melibatkan semua dokumen latih di korpus. Maka dilakukan modifikasi pembobotan dengan menggunakan tf-idf, yakni *term frequency* yang dikalikan *inverse document frequency* (idf). Idf menghitung bobot token/kolokasi berdasarkan jumlah total dokumen di korpus (N) dibagi dengan document frequency (df) yang menunjukkan jumlah dokumen sebuah token/kolokasi muncul. IDF dikomputasi menggunakan persamaan 1.

$$idf_t = \log \frac{N}{df_t} \quad (1)$$

Pembobotan tf-idf dipilih dengan mempertimbangkan studi yang dilakukan oleh Shahmirzadi, Lugowski dan Younge [15]yang menyatakan bahwa pembobotan ini masih unggul dibandingkan metode pembobotan lainnya. Setelah pembobotan tf-idf, maka proses pemodelan topik dijalankan sama seperti proses yang ada di Lexikat seperti yang dijelaskan sebelumnya. Tentu saja klasterisasi token/kolokasi yang dhasilkan berbeda seperti yang ditunjukkan oleh Gambar 8, dimana kolom kiri menunjukkan sebuah topik yang dihasilkan dengan pembobotan tf-idf, sedangkan kolom kanan adalah topik yang dihasilkan dengan pembobotan tf saja.

TFIDF Clustering	Normal Clustering
'parents',	'parents',
'neighbors',	'neighbors',
'grandfather',	'ancestors',
'ancestors',	'family',
'relatives',	'relatives',
'neighbor',	'friends',
'friends',	'neighbor',
'family',	'grandfather',
'ambulance',	'bus',
'bus',	'ambulance',
'an ambulance',	'an ambulance',
'karabagh',	'told',
'baku',	'told me',
'apartment',	'answered',
'basement',	'told him',
'upstairs',	'loved',
'house']	'always loved',
	'dreamed',
	'forgot',
	'approached',
	'met',
	'armenians',
	'baku'

Gambar 8 perbedaan sebuah topic yang dihasilkan dengan pembobotan local (tf) dan global (tf-idf).

Kumpulan topik inilah yang disimpan sebagai index dalam inverted index sedangkan posting list berisi ID dokumen dan bobot tiap elemen topik.

E. Pemrosesan Dokumen Uji

Dalam proses temu kembali dokumen termirip, maka dibutuhkan antarmuka untuk menerima dokumen uji sebagai input. Antarmuka dibangun berbasis web dengan memanfaatkan pustaka ReactJS. Dokumen uji sebagai input akan diteruskan oleh API endpoint untuk diproses. Tahap prapemrosesan dokumen uji ini mirip dengan tahapan dalam pembuatan *inverted index.*

Untuk mendapatkan topik dari dokumen uji, maka sistem temu kembali yang dibangun ini mengintegrasikan API Lexikat yang menerima teks yang sudah dinormalisasi tersebut. Proses pembobotan yang sama yakni tf-idf juga diterapkan setelah proses tokenisasi dan pembentukan kolokasi. Proses klasterisasi token dan kolokasipun

dilakukan di Lexikat. Luaran klasterisasi token ini diproses kembali dengan mengabungkan beberapa topic yang saling terkait, dan menghapus elemen topik yang telah digabungkan seperti yang ditunjukkan di Gambar 5. Luaran kategori kata inilah yang diperhitungkan sebagai topik.

Berbeda dari proses pembangunan Index, kumpulan topik ini diformulasikan sebagai kueri dokumen uji. Tiap elemen topik termasuk kolokasi diperlakukan sebagai sebuah token individual. Perankingan topik berdasarkan bobot kategori token dilakukan untuk mempermudah proses pemilihan jumlah topik yang akan dijadikan kueri dokumen uji. Jika hanya 1 atau 2 topik yang digunakan sebagai kueri, maka akan dipilih 2 topik dengan urutan tertinggi dan dua topik ini dilebur menjadi sebuah kueri.

Pengukuran kemiripan dokumen uji yang direpresentasikan oleh kueri dengan dokumen sumber dilakukan dengan memanfaatkan *inverted index* yang telah terbangun. Saat token kueri ditemukan dalam sebuah dokumen, maka nilai vektornya, dalam hal ini tf-idf akan diambil dan ditampung untuk perhitungan kemiripan dokumen dengan menggunakan ukuran kemiripan *Cosine* seperti yang ditunjukkan ditunjukkan di Persamaan 2. Penghitungan kemiripan ini dilakukan di setiap dokumen sumber yang mengandung salah satu, beberapa, atau semua token di kueri.

$$\cos(\vec{q},\vec{d}) = \frac{\vec{q}.\vec{d}}{\|\vec{q}\|\|\vec{d}\|} = \frac{\sum_{i=1}^{|v|}\vec{q}_i.\vec{d}_i}{\sqrt{\sum_{i=1}^{|v|}\vec{q}_i}.\sqrt{\sum_{i=1}^{|v|}\vec{d}_i}} \quad (2)$$

q_i adalah bobot tf-idf token i di kueri dan d_i adalah bobot tf-idf token i di dokumen sumber yang terindeks, sedangkan $\|\vec{q}\|$ merujuk pada panjang vector kueri yang telah dinormalkan yakni akar dari penjumlahan nilai vector tiap token di kueri. Demikian juga berlaku untuk $\|\vec{d}\|$ yang merujuk ke normalisasi panjang vektor term di dokumen d.

IV. Evaluasi sistem dan pembahasan

Untuk melakukan pengujian sistem atau eksperimen, maka dokumen yang ada di korpus 20 Newsgroup dipartisi menjadi dua dengan rasio sekitar 10% digunakan sebagai dokumen uji dari tiap klaster dokumen dan 90% sebagai dokumen sumber yang diindeks di *Inverted Index*. Penentuan rasio dan dokumen uji ini mengikuti partisi yang dibuat oleh korpus 20 Newsgroups untuk memudahkan pengukurannya karena label dari dokumen uji dan sumber sudah tersedia disana. Tabel 2 menujukkan 6 contoh dari 19 kategori yang ada di korpus 20 Newsgroups beserta jumlah dokumen uji dan dokumen sumber.

Metrik evaluasi yang digunakan adalah presisi, *recall*, F-1 dan Break-Even Point dengan penjeleasan sebagai berikut:

- **Presisi dan *Recall*** dihitung berdasarkan luaran sistem yang dibandingkan dengan label dokumen uji berdasarkan kategorinya. Relevansi kueri dari dokumen uji dianggap valid, jika dokumen yang ditemukan memiliki label kategori yang sama dengan dokumen uji. Label kategori ini didapatkan langsung dari korpus 20 Newsgroups. Metriks yang digunakan anda Macro-Averaged Precision (MAP) dan Macro-Averaged Recall (MAR) yang dikomputasi dengan menggunakan persamaan 3 dan 4.

$$MAP = \frac{1}{n}\sum_{i=1}^{n} Precision(d_i) \quad (3)$$

$$MAR = \frac{1}{n}\sum_{i=1}^{n} Recall(d_i) \quad (4)$$

- **F-1** merupakan rata-rata yang seimbang antara presisi dan *Recall* dan dikomputasi dengan Persamaan 5, dimana P merujuk pada presisi dan R pada *Recall*. Rumus F1 yang digunakan juga Macro-averaged F1 (MAF).

$$MAF = \frac{1}{n}\sum_{i=1}^{n} F1(d_i) \quad (5)$$

- **Break-Even Point** (BEP) merupakan nilai yang didapatkan saat Presisi dan *Recall* memiliki nilai yang sama dan didapatkan dengan cara meng-interpolasikan nilai presisi dan *Recall* terdekat.

TABEL 2. CONTOH RASIO DOKUMEN UJI DAN DOKUMEN SUMBER DARI KORPUS 20 NEWSGROUPS

Kategori	Dokumen Uji	Dokumen Sumber	Total Dokumen
sci_crypt	35	301	336
comp_graphics	13	128	141
rec_motorcycles	20	241	261
rec_sport_hockey	18	254	272
sci_med	31	239	270
.....	.....	.....	....
sci_space	26	242	268
Total Dokumen	474	4266	4740

A. Skenario Pengujian

Dalam pengujian sistem, maka ada 2 hal yang dicoba dieksperimenkan yakni di tahap praproses teks dan pembobotan. Skenario eksperimen di tahap praproses teks adalah:

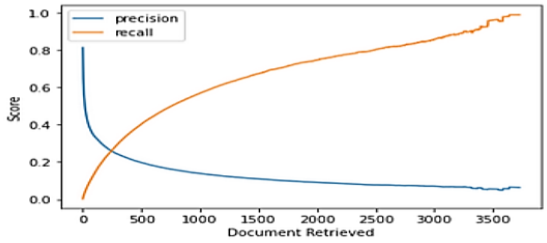
- ***Skenario* preprocessed** adalah data dipraproses menggunakan lowercase dan penghapusan email, angka dan karakter spesial.
- **Skenario preprocessed_lemma** adalah data dinormalkan melalui lowercase, penghapusan email, angka, dan karakter spesial, dan lemmatisasi.

- **Skenario preprocessed_stopwords** merupakan skenario proprocessed yang ditambahi proses penghapusan kolokasi stopwords.
- **Skenario preprocessed_lemma_stopwords** adalah data dipraproses menggunakan lowercase, penghapusan email, angka dan karakter spesial, lemmatisasi dan penghapusan kolokasi stopwords.
- **Skenario preprocessed_stopwords_lemma** adalah data dinormalkan melalui lowercase, penghapusan email, angka, dan karakter spesial, dan penghapusan kolokasi stopwords. Perbedaan skenario ini dari skenario sebelumnya adalah fase lematisasi yang diterapkan setelah proses pemodelan topik. Dengan kata lain, elemen topik akan dilakukan lematisasi sehingga nilai tf dan tf-idf akan berbeda dengan topik yang dihasilkan dari seknario preprocessed_lemma_ stopwords.

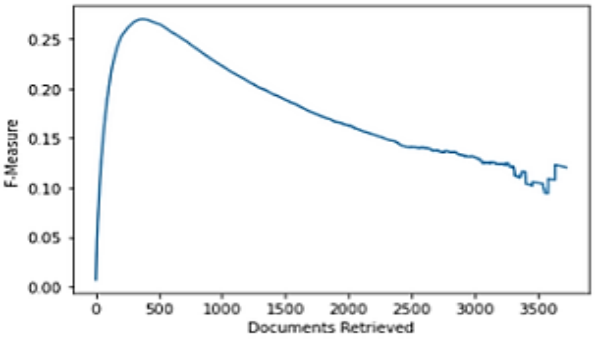
Sedangkan skenario pembobotan term dibedakan menjadi dua yakni pembobotan murni Lexikat dan pembobotan tf-idf. Dua kategori pembobotan ini digabungkan dengan skenario praproses sehingga ada 10 eksperimen.

Sistem yang dibangun dirancang sedemikian rupa untuk menerima input lebih dari 1, sehingga pengguna bisa memilih 2 atau 3 dokumen sekalipus sebagai dokumen uji (input) atau seluruh dokumen uji yang ada di korpus. Antarmuka dari sistem yang dibangun ditampilkan di lampiran di akhir artikel ini. Dalam pengujian, kami langsung menginputkan semua dokumen uji yang berjumlah 474 (cf. Tabel 2). Tiap dokumen uji ini dibandingkan dengan dokumen sumber yang berjumlah 4.266 dan yang telah diindeks. kemudian MAP, MAR dan MAF dikomputasi dalam satu kali putaran.

Eksperimen yang dilakukan tidak menerapkan nilai batas ambang (threshold) untuk memotong jumlah dokumen dengan nilai kemiripan Cosine tertinggi. Hal ini mempengaruhi nilai MAP, MAR dan MAF yang tidak terlalu tinggi karena kami ingin melihat jumlah dokumen untuk menghasilkan nilai MAF dan BEP terbaik. Setelah 5 skenario praproses data dijalankan, hasil MAP, MAR, MAF, dan BEP tidaklah terpaut jauh. Nilai tertinggi keempat MAP, MAR dan F1 dicapai oleh skenario *preprocessed_stopwords* dimana skor F1 mencapai 0.2697 dan grafiknya ditampilkan di Gambar 9.



(a)



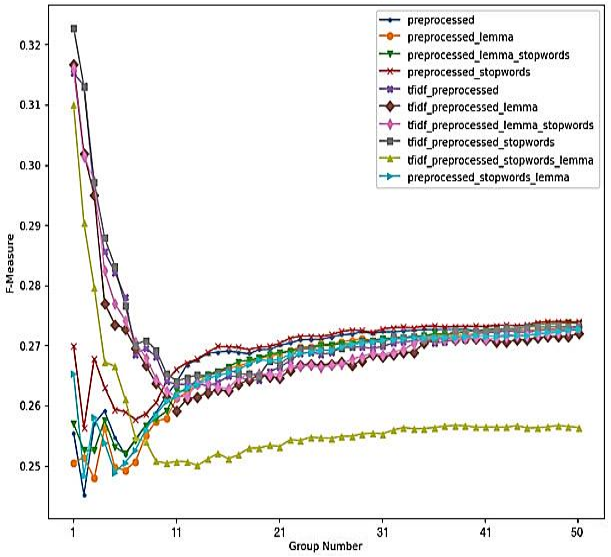
(b)

Gambar 9 Visualisasi nilai interpolasi MAP dan MAR dalam skenario *preprocessed_stopwords* (a), sedangkan (b) menampilkan nilai MAF dalam skenario yang sama.

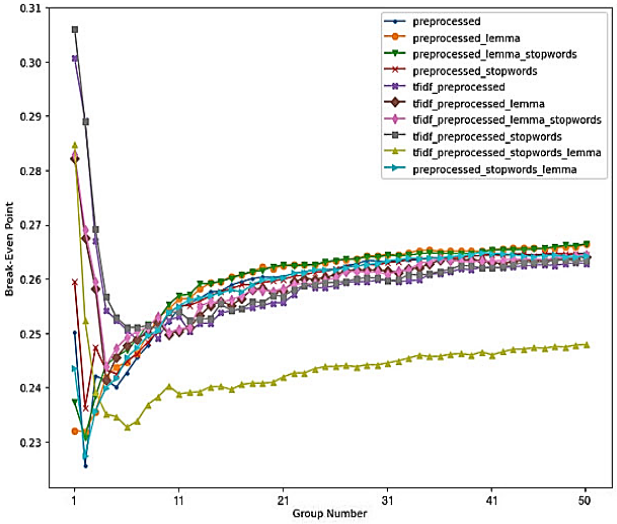
Eksperimen demgan hasil yang ditunjukkan di Gambar 9 tersebut menggunakan 20 topik sebagai queri dan jumlah token di tiap topik berbeda-beda seperti yang ditunjukkan di Gambar 7. Dengan menggunakan jumlah topik yang sama, maka nilail BEP tertinggi diraih oleh skenario *preprocessed_lemma_stopwords* yang mencapai 0.2622. Jumlah dokumen yang dibutuhkan untuk mencapai nilai BEP tertinggi berkisar antara 243-244 dokumen sedangkan untuk mencapai skor F1 tertinggi untuk masing-masing skenario adalah:

- *Preprocessed*: Skor F1 mencapai puncak tertinggi di dokumen ke 358.
- *Preprocessed_lemma:* setelah dokumen ke-386 maka skor F1 menurun.
- *Preprocessed_stopwords:* skor F1 mencapai puncak tertinggi di dokumen ke 366.
- *Preprocessed_lemma_stopwords:* di dokumen ke-396 diperoleh nilai F1 tertinggi.
- *Preprocessed_stopwords_lemma*: setelah dokumen ke-350 maka nilai F1 mengalami penurunan.

Dalam eksperimen selanjutnya, digunakan 50 topik sebagai kueri dari tiap dokumen uji dengan pembobotan ttf-idf. Nilai skor F1 di tiap skenario pengujian ditunjukkan di Gambar 10. Jika Gambar 10 ini diamati, maka skenario preprocessed_stopwords tetap memiliki nilai F1 yang tertinggi saat menggunakan lebih dari 11 topik. Sedangkan skor F1 tertinggi dengan penggunaan kurang dari atau sama dengan 10 topik dicapai oleh skenario *tfidf_preprocessed_ stopwords.* Sedangkan Gambar 11 menunjukkan nilai BEP dari keseluruhan skenario pengujian dengan 50 topik sebagai kueri. Kecenderungan yang sama juga ditunjukkan oleh nilai BEP dan jika menggunakan lebih dari 11 topik sebagai kueri, maka nilai BEP tertinggi dicapai oleh *preprocessed_lemma_stopwords* dan p*reprocessed_lemma.*



Gambar 10 Visualisasi nilai F1 hasil eksperimen dari keseluruhan skenario dengan 50 topik sebagai kueri.



Gambar 11 Visualisasi nilai BEP hasil eksperimen dari keseluruhan skenario dengan 50 topik sebagai kueri.

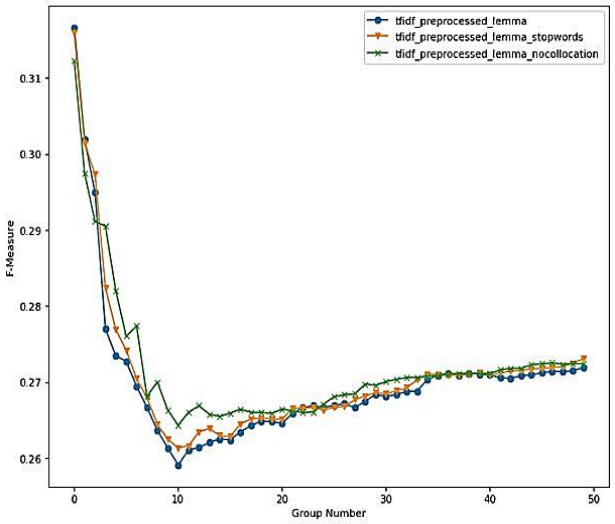
Eksperimen lainnya dilakukan tanpa generasi topik dari dokumen uji namun memanfaatkan semua *term* sebagai fitur dokumen uji. Nilai F1 dan BEP dari eksperimen ini ditunjukkan di Tabel 3.

TABEL 3 SKOR F1 DAN NILAI BEP SKENARIO PENGUJIAN DENGAN MENGGUNAKAN SMEUA FITUR DOKUMEN UJI SEBAGAI KUERI

Skenario pengujian	**F-1**	**BEP**
Preprocessed	0.2672	0.2694
Preprocessed_lemma	0.2774	0.2701
Preprocessed_stopwords	0.2761	0.2685
Preprocessed_lemma_stopwords	0.2765	0.2696
Preprocessed_stopwords_lemma	0.277	0.2695
Tfidf_preprocessed	0.2761	0.2683
Tfidf_preprocessed_lemma	0.2763	0.2696
Tfidf_preprocessed_stopwords	0.2773	0.2694
Tfidf_preprocessed_lemma_stopwords	0.2775	0.2702
Tfidf_preprocessed_stopwords_lemma	0.2619	0.2536

Tabel 3 menunjukkan bahwa nilai F1 dan BEP antara skenario yang menggunakan semua fitur dokumen uji dibanding dengan penggunaan 50 topik tertinggi tidaklah terpaut jauh. Ini dikarenakan 50 topik pertama dalam dokumen pendek hampir mencakup semua term. Dari sisi waktu pemrosesan, saat 1 topik digunakan, maka hanya membutuhkan 3 detik sedangkan untuk memproses 50 topik sebagai kueri membutuhkan waktu 3 menit. Waktu proses bagi penggunaan semua fitur sebagai kueri mencapai 5 menit.

Kueri yang terbentuk dari topik di pengujian sebelumnya mengandung kolokasi yang diperlakukan sebagai sebuah string tunggal. Untuk itu, eksperimen terakhir yang dilakukan adalah dengan menghilangkan semua kolokasi yang muncul di tiap topik. Gambar 12 menampilkan skor F1 dari skenario pengujian dengan pembobotan tf-idf sebagai sampel.



Gambar 12 Visualisasi hasil uji dengan skenario tanpa kolokasi yang menggunakan pembobotan tf-idf.

Gambar 12 hanya menampilkan 2 skenario pengujian yang menggunakan kolokasi dengan pembobotan tfidf yang mencapai nilai F1 tertinggi dibandingkan dengan skenario yang sama namun tanpa kolokasi dengan 50 topik sebagai kueri. Sekalipun hasilnya tidak menunjukkan perbedaan yang signifikan, namun nilai F1 dari skenario tfidf_preprocessed_lemma tanpa kolokasi lebih tinggi. Fenomena yang sama juga terjadi pada nilai BEP dengan skenario yang sama. Penjelasannya sederhana saja, kolokasi yang diperlakukan sebagai satu string tunggal membuat string ini sangat spesifik sehingga saat dihilangkan skor kemiripan *Cosine* juga meningkat dan ini mempengarui nilai MAP dan MAR yang berimbas pada MAF.

B. Pembahasan

Dari 11 skenario eksperimen yang diujikan, nilai *Macro-Averaged F-score* (MAF) tertinggi berkisar antara 0.32 – 0.33 saat menggunakan 1 topik sebagai kueri. Sedangkan jumlah dokumen minimal yang diperlukan untuk mencapai nilai BEP tertinggi adalah 243. Jumlah dokumen ini bisa dijadikan rekomendasi sebagai nilai ambang (*threshold*) untuk penyaringan dokumen yang dievaluasi. Dan jika ini diterapkan dalam proses *filtering*, maka bisa dipastikan bahwa nilai MAF akan meningkat secara otomatis karena nilai presisinya naik tajam.

Hasil eksperimen ini menunjukkan bahwa Pemodelan topik dengan pembobotan TF-IDF menghasilkan performa yang lebih baik dari pemodelan topik pembobotan TF jika jumlah topik yang digunakan sebagai *query* kurang dari 10. Jumlah topik yang digunakan sebagai kueri menentukan spektrum kemiripan dokumen yang ditemukan. Jumlah topik yang rendah menemukan dokumen dengan relasi topik di beberapa bagian dokumen saja, sedangkan semakin tinggi jumlah topik yang digunakan akan menghasilkan dokumen yang memiliki kemiripan di tingkat leksikal dan sintaksis. Penggunaan Pemodelan topik untuk temu kembali sumber dokumen termirip dengan kueri ini menghasilkan nilai MAF yang tidak jauh berbeda dengan penggunaan pembobotan lokal dalam pembentukan kueri yang diterapkan oleh Haryanto dkk [14], dimana nilai MAF tertingginya mencapai 0.418.

V. KESIMPULAN DAN SARAN

Usaha untuk menemukan dokumen termirip dengan dokuumen kueri dilakukan dengan membangun sistem temu kembali yang menggunakan topik untuk proses pengindeksan maupun pembentukan kueri dokumen input. Topik dihasilkan dari sistem Pemodelan Topik Lexikat yang diimpor sebagai salah satu modul dalam sistem temu kembali kami. Untuk mengevaluasi kinerja sistem, maka korpus 20 newsgroups digunakan baik sebagai dokumen sumber maupun dokumen uji. Sedangkan metriks ukuran yang digunakan adalah MAP, MAR , MAF, dan nilai BEP. Skor MAF menunjukkan kinerja sistem terhadap seluruh dokumen uji yang berjumlah 474 dokumen dan nilai BEP menunjukkan jumlah dokumen yang diperlukan untuk mencapai skor MAP dan MAR yang sama.

Salah satu kelemahan dari penelitian ini terletak pada penggunaan korpus yang belum menyediakan label pada bagian dokumen yang memiliki kemiripan di tingkat sintaksis, sehingga evaluasi dilakukan secara manual saat mengecek luaran sistem yang menggunakan 50 topik sebagai kueri. Selain itu, belum diberlakukan penyaringan dokumen sumber berdasarakan nilai kesamaan Cosine membuat nilai MAF cukup rendah karena semua dokumen yang hanya mengandung 1 atau dua elemen dari salah satu topik juga disertakan dalam evaluasi.

Agar dokumen dengan kemiripan di tingkat sintaksis bahkan semantik bisa ditemukan, maka diperlukan pembuatan dokumen uji hasil simulasi atau digenerasikan secara algoritmik dari dokumen sumber yang ada. Kemudian pelabelanpun dibuat lebih halus dengan kemiripan bertingkat yakni di tingkat topik (leksikal), sintaksis dan semantik.

REFERENSI

[1] R. Baeza-Yates dan B. Ribeiro-neto, Modern Information Retrieval, second penyunt., Addison Wesley, 2011.

[2] C. Manning, P. Raghavan dan P. Schuetze, Introduction to Information Retrieval, Cambridge: Cambridge University Press, 2009.

[3] J. Zobel, "What We Talk About When We Talk About Information Retrieval," *ACM SIGIR Forum,* vol. 51, no. 3, pp. 18-26, 2017.

[4] L. D. Krisnawati dan K. U. Schulz, "Significant Word-based Text Alignment for Text Reuse Detection," dalam *Int'l Conference on Research & Innovation in Computer, Electronics and Manufacturing Engineering (RICEME-17),* Bali, Indonesia, 2017.

[5] J. S. Wang, "Topic Modeling: A Complete Introductory Guide," dalam *NLP Day*, Austin, Texas, 2017.

[6] S. Yusuf, M. Fauzi dan K. Brata, "Sistem Temu Kembali Informasi Pasal-Pasal KUHP (Kitab Undang-Undang Hukum Pidana) Berbasis Android Menggunakan Metode Synonym Recognition dan Cosine Similarity," *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer,* vol. 2, no. 2, pp. 838-847, 2017.

[7] M. A. Lamongi, R. Munir dan A. Angdresey, "Aplikasi Sistem Temu Kembali Informasi di Dalam Alkitab Menggunakan Model Ruang Vektor.," Repository Uniika De La Salle , Manado, 2019.

[8] S. Gunawan, L. D. Krisnawati dan A. R. Chrismanto, "Analisis Fiitur Stilometri dan Strategi Segmentasi pada Sistem Deteksi Plagiasi Intrinsik Teks," *Rekayasa Sistem dan Teknologi Informasi (RESTI),* vol. 4, no. 5, pp. 988-997, 2020.

[9] B. Boenninghoff, S. Hessler, D. Kolossa dan R. Nickel, "Explainable Author Verification in Social Media via Attention-based Similarity Learning," dalam *International Conference on Big Data*, 2019.

[10] L. D. Krisnawati, "Plagiarism detection for Indonesian texts," Elektronishe Hochschulschriften, Ludwig-Maximilan Univeristaet, Munich, 2016.

[11] A. Onan, S. Korukoglu dan H. Bulut, "LDA-Based Topic Modelling in Text Sentiment Classification: An Empirical Analysis," *Int. Journal Computational Linguistics Application,* vol. 7, pp. 101-119, 2016.

[12] H. Gong, F. You, X. Guan, Y. Cao dan L. Lai, "Application of LDA Topic Model in E-Mail Subject Classification," dalam *Intl. Conf. on Transportation and Logistics, Information and Communication, Smart City*, 2018.

[13] I. G. Anugrah dan H. Rosyid, "Penerapan Information Retrieval Menggunakan Pemodelan Topik Pada Deskripsi Portal Multimedia," *Anugrah, I. G., & Rosyid, H. (2019). Penerapan Information Retrieval MenggunakanJurnal Nasional Komputasi dan Teknologi Informasi.,* vol. 2, no. 1, pp. 48-54, 2019.

[14] N. C. Haryanto, L. D. Krisnawati dan A. R. Chrismanto, "Temu Kembali Dokumen Sumber Rujukan dalam Sistem Daur Ulang Teks," *Jurnal Teknologi dan Sistem Komputer,* vol. 8, no. 2, pp. 140-149, 2020.

[15] O. Shahmirzadi, A. Lugowski dan K. Younge, "Text Similarity in Vector Space Models: A Comparative Study," *CoRR,* vol. abs/1810.00664, 2018.

Lampiran 1. Antarmuka sistem temu Kembali yang menggunakan Pemodelan Topik. Antarmuka ini menunjukkan saat dokumen uji dimasukkan dan tombol Most Similar di menu kiri atas diklik, maka akan menampilkan fitur dari dokumen uji, fitur dokumen sumber dan fitur yang sama, sekaligus hasil peengujiannya.