IMPLEMENTASI *LATENT DIRICHLET ALLOCATION* (LDA) DAN K-*NEAREST NEIGHBORS* (KNN) PADA SISTEM REKOMENDASI JURNAL TERINDEKS GARUDA

LAPORAN TUGAS AKHIR

Laporan ini Disusun untuk Memenuhi Salah Satu
Syarat Memperoleh Gelar Sarjana Strata 1 (S1) pada
Program Studi Teknik Informatika Fakultas Teknologi
Industri Universitas Islam Sultan Agung Semarang

DISUSUN OLEH:

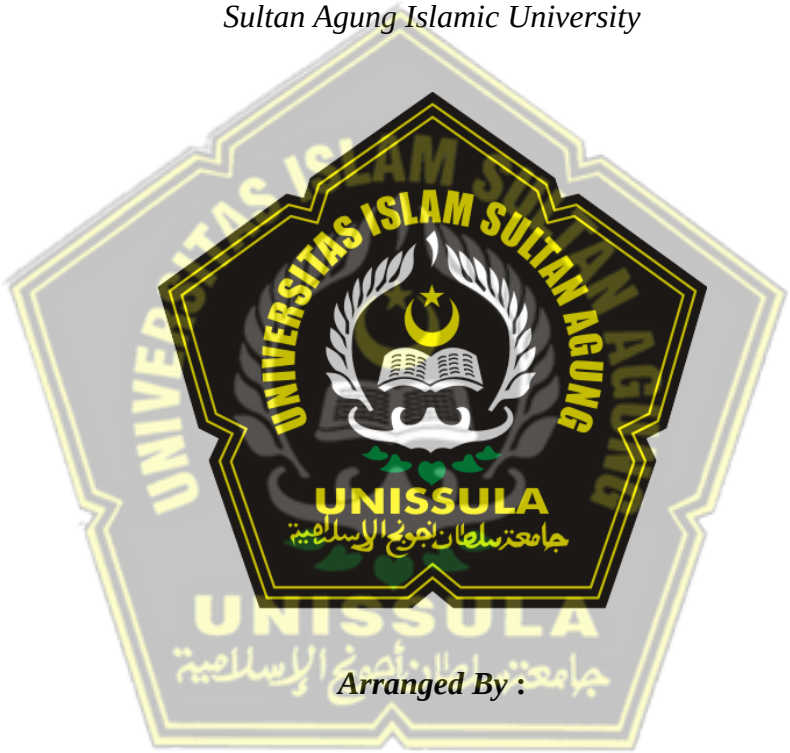
ANISATUZZUMARA
NIM 32602000015

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVRSITAS ISLAM SULTAN AGUNG
SEMARANG
2024

FINAL PROJECT

LATENT DIRICHLET ALLOCATION (LDA) AND K-*NEAREST NEIGHBORS* (KNN) *IMPLEMENTATION ON RECOMMENDATION SYSTEM GARUDA INDEXED JOURNAL*

Proposed to complete the requirement to obtain a bachelor's degree (S1) at Informatics Engineering Departement of Industrial Technology Faculty Sultan Agung Islamic University



Arranged By :

ANISATUZZUMARA

NIM 32602000015

MAJORING OF INFORMATICS ENGINEERING

INDUSTRIAL TECHNOLOGY FACULTY

SULTAN AGUNG ISLAMIC UNIVERSITY

SEMARANG

2024

LEMBAR PENGESAHAN PEMBIMBING

Laporan Tugas Akhir dengan judul **"Implementasi *Latent Dirichlet Allocation* (LDA) dan K-*Nearest Neighbors* (KNN) Pada Sistem Rekomendasi Jurnal Terindeks GARUDA"** ini disusun oleh :

Nama : Anisatuzzumara

NIM : 32602000015

Program Studi : Teknik Informatika

Telah disahkan oleh dosen pembimbing pada :

Hari : Senin

Tanggal : 4 Maret 2024

Mengesahkan,

Pembimbing I	Pembimbing II
Sam Farisa Chaerul Haviana, S.T., M.Kom	Badiyah, S.T., M.Kom
NIDN. 0628028602	NIDN. 0619018701

Mengetahui,

Ketua Program Studi Teknik Informatika

Fakultas Teknologi Industri

Universitas Islam Sultan Agung

Dr. Muljono, M.Eng

NIDN. 0626066601

LEMBAR PENGESAHAN PENGUJI

Laporan tugas akhir dengan judul "Implementasi *Latent Dirichlet Allocation* (LDA) Dan K-*Nearest Neighbors* (KNN) Pada Sistem Rekomendasi Jurnal Terindeks GARUDA" ini telah dipertahankan di depan dosen penguji Tugas Akhir pada :

Hari : Jumat

Tanggal : 1 Maret 2024

TIM PENGUJI

Ketua Penguji	Anggota I
Ghufron, S.T., M.Kom	Bagus Satrio WP, S.Kom., M.Cs
NIDN. 0602079005	NIDN. 1027118801

SURAT PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan dibawah ini :

Nama : Anisatuzzumara
NIM : 32602000015
Judul Tugas Akhir : Implementasi *Latent Dirichlet Allocation* (LDA) dan K-*Nearest Neighbors* (KNN) Pada Sistem Rekomendasi Jurnal Terindeks GARUDA

Dengan bahwa ini saya menyatakan bahwa judul dan isi Tugas Akhir yang saya buat dalam rangka menyelesaikan Pendidikan Strata Satu (S1) Teknik Informatika tersebut adalah asli dan belum pernah diangkat, ditulis ataupun dipublikasikan oleh siapapun baik keseluruhan maupun sebagian, kecuali yang secara tertulis diacu dalam naskah ini dan disebutkan dalam daftar pustaka, dan apbila di kemudian hari ternyata terbukti bahwa judul Tugas Akhir tersebut pernah diangkat, ditulis ataupun dipublikasikan, maka saya bersedia dikenakan sanksi akademis. Demikian surat pernyataan ini saya buat dengan sadar dan penuh tanggung jawab.

Semarang, 17 Februari 2024

Yang Menyatakan,

Anisatuzzumara

KATA PENGANTAR

Dengan mengucapkan syukur alhamdulillah atas kehadirat Allah SWT yang telah memberikan rahmat dan karunianya kepada penulis, sehingga dapat menyelesaikan Tugas Akhir dengan judul "Implementasi *Latent Dirichlet Allocation* (LDA) dan *K-Narest Neighbors* (KNN) Pada Sistem Rekomndasi Jurnal Terindeks GARUDA" ini untuk memenuhi salah satu syarat menyelesaikan studi serta dalam rangka memperoleh gelar sarjana (S-1) pada program Studi Teknik Informatika Fakultas Teknologi Industri Universitas Islam Sultan Agung Semarang.

Tugas Akhir ini disusun dan dibuat dengan adanya bantuan dari berbagai pihak, materi maupun teknis, oleh karena itu saya selaku penulis mengucapkan terima kasih kepada :

1. Rektor UNISSULA Bapak Prof. Dr. H. Gunarto, S.H., M.H yang mengizinkan penulis menimba ilmu dikampus ini.
2. Dekan Fakultas Teknologi Industri Ibu Dr. Ir. Novi Marlyana, S.T., M.T., IPU.,ASEAN. Eng
3. Dosen Pembimbing I penulis Bapak Sam Farisa Chaerul Haviana, S.T., M.Kom yang telah meluangkan waktu dan memberi ilmu.
4. Dosen Pembimbing II penulis Ibu Badie'ah, S.T., M.Kom yang telah memberikan banyak nasehat dan saran.
5. Orang tua penulis yang telah mengizinkan untuk menyelesaikan laporan ini serta memberikan dukungan dan doa agar tugas akhir ini dapat berjalan dengan lancar.
6. Dan kepada semua pihak yang tidak dapat saya sebutkan satu persatu.

Dengan segala kerendahan hati, penulis menyadari masaih terdapat banyak kekurangan dari segi kualitas atau kuantitas maupun dari ilmu pengetahuan dalam penyusunan laporan, sehingga penulis mengharapkan adanya saran dan kritikan yang bersifat membangun demi kesempurnaan laporan ini di masa mendatang.

Semarang, 27 Februari 2024

Anisatuzzumara

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

ABSTRAK

Adanya peningkatan jumlah jurnal dari tahun ke tahun, membuat para pihak peneliti dan profesional membutuhkan alat yang efektif guna menemukan jurnal yang relevan dengan riset yang sedang dilakukan. *Topic modelling* adalah jenis model statistik yang tidak memiliki pelabelan teks dengan tujuan untuk menemukan topik atau tema secara garis besar dari sekumpulan (korpus). LDA merupakan kumpulan dokumen sebagai topik campuran yang berisi kata-kata dengan probabilitas tertentu. Implementasi algoritma LDA mampu untuk mendeteksi topik yang ada pada suatu koleksi dokumen beserta besarnya kemunculan topik tersebut pada koleksi dokumen. K-*Nearest Neighbors* (KNN) adalah algoritma pembelajaran mesin yang mudah namun produktif, ini efektif untuk klasifikasi dan regresi. Implementasi metode *Latent Dirichlet Allocation* (LDA) dan K-*Nearest Neighbors* (KNN) pada sistem rekomendasi jurnal terindeks GARUDA ini berjalan dengan baik saat memodelkan topik dan memetakannya kedalam beberapa jenis *topic_id* dan *doc_id*.

Kata Kunci: *Topic Modelling*, LDA, K-NN, Sistem Rekomendasi, GARUDA

ABSTRACT

There is an increase in the number of journals from year to year, making researchers and professionals need effective tools to find journales that are relevant to the ongoing research. Topic Modelling is a type of statistical model that does not have text labels with the aim of finding topics or themes in a large line of a set (corpus). LDA is a collection of documents as a mixed topic that contains words with a certain probability. The LDA implementations algorithm is capable of detecting topics that exist in a document collection as well as the extent to which such topics occur in the document collection. K-Nearest Neighbors (KNN) is a simple yet productive machine learning algorithm, it's effective for classification and regression. Implementation of Latent Dirichlet Allocation (LDA) and K-Nearest Neighbors (KNN) methods on GARUDA's indexed journal recommendation system This works well when modeling topics and taking them into some type of topic_id and doc_id.

Keywords: Topic Modelling, LDA, K-NN, System Recommendation, GARUDA.

BAB I
PENDAHULUAN

1.1 Latar Belakang

Adanya peningkatan jumlah jurnal dari tahun ke tahun, membuat para peneliti termasuk akademisi yakni dosen dan mahasiswa membutuhkan alat yang efektif guna menemukan jurnal yang relevan dengan riset yang sedang dilakukan. Seringkali dalam mencari jurnal yang relevan, para peneliti harus melakukan pencarian di dalam mesin pencari dalam kurun waktu yang relatif lama dikarenakan jurnal - jurnal yang direkomendasikan kurang sesuai dengan topik yang dicari. Disamping itu, banyaknya jurnal-jurnal yang *publish* dengan nama topik penelitian yang sejenis akan tetapi model penelitian yang berbeda sering kali mengakibatkan pencarian jurnal terindeks menjadi lebih sulit. Maka dari itu, tentu diperlukan sistem algoritma yang dapat membantu mengurangi tingkat waktu yang diperlukan dan nilai akurasi ketepatan dalam pencarian jurnal terindeks sehingga, jurnal rekomendasi yang dicari dapat didapatkan dengan lebih cepat dan lebih sesuai dengan keinginan peneliti.

Faktor utama penerimaan artikel dalam publikasi adalah relevansi artikel terhadap topik pada jurnal yang dituju memerlukan adanya sistem klasifikasi artikel jurnal untuk membantu peneliti dan professional dalam memilih jurnal untuk artikelnya (Ristanti *dkk.*, 2019). Jurnal ilmiah sendiri berisi artikel-artikel yang memiliki banyak topik dan disiplin ilmiah yang beragam, sehingga sulit bagi peneliti untuk menemukan jurnal yang paling sesuai dengan minat mereka. Peluang untuk mengembangkan sistem rekomendasi yang lebih canggih dan efektif telah muncul sebagai hasil dari kemajuan dalam teknologi informasi untuk membantu peneliti menemukan jurnal yang tepat.

Salah satu sistem algoritma yang dapat diimpelementasikan dalam hal ini adalah LDA (*Latent Dirichlet Allocation*). LDA merupakan kumpulan dokumen sebagai topik campuran yang berisi kata-kata dengan probabilitas tertentu. Algoritma LDA mampu untuk mendeteksi topik yang ada pada suatu koleksi dokumen beserta besarnya kemunculan topik tersebut pada koleksi dokumen

tertentu (Kurniawan R *dkk*., 2022). LDA dari segi definisi adalah model probabilistik generatif dan algoritma *unsupervised learning* dimana dokumen direpresentasikan sebagai campuran acak topik-topik. Dimana topik- topik tertentu dicirikan oleh distribusi kata-kata dari dokumen- dokumen yang menjadi korpus atau kumpulan teks alami, baik bahasa lisan maupun bahasa tulis, yang disusun secara sistematis (Novarian *dkk*., 2023).

Sistem LDA dipilih pada penelitian ini dikarenakan *Latent Dirichlet Allocation* (LDA) memiliki keunggulan yaitu bisa mengklasterkan data yang jumlahnya besar dibandingkan beberapa tipe metode pemodelan topik yang lain serta dapat diimplementasikan untuk mengidentifikasi topik dalam jurnal ilmiah, klasifikasi, dan pengelompokan (Oktafiandi, 2023). Selain itu kelebihan metode LDA adalah dapat mengekstrak topik secara akurat pada kumpulan data yang cukup besar sehingga potensi diperolehnya jurnal dengan tema dan topik yang sesuai keinginan peneliti. Dengan menggunakan sistem LDA pada penelitian ini akan diperoleh konklusi apakah implementasi LDA mampu mempersingkat waktu pencarian jurnal serta nilai ketepatan jurnal yang dipilih (Purwitasari *dkk.*, 2021).

Maka dari itu, sistem ini dapat diimplementasikan didalam pengembangan sistem rekomendasi terkait jurnal yakni sistem rekomendasi jurnal yang mengambil dataset dari salah satu *platform online*. Pada penelitian ini *platform online* jurnal yang dipilih yaitu Garba Rujukan Digital (GARUDA) yang menyediakan akses ke banyaknya jurnal-jurnal ilmiah maupun non ilmiah. GARUDA dipilih karena platform ini menyediakan beragam jurnal yang berisi artikel-artikel yang bersumber dari berbagai peneliti yang mempunyai tingkat akreditasi rendah maupun tinggi serta jurnal- jurnal internasional terpublikasi (Sa'adah, 2022).

Oleh karena itu, penelitian tugas akhir ini dibuat dengan menggunakan untuk mengetahui seberapa tinggi nilai relevan pencarian jurnal terindeks GARUDA dari penggunaan metode LDA dibandingkan dengan pencarian jurnal tanpa disertai dengan implementasi LDA.

1.2 Rumusan Masalah

Berdasarkan latar belakang tersebut, penulis mengidentifikasi permasalahan yang akan dibahas yaitu:

1. Mengetahui bagaimana metode LDA mengidentifikasi dan memetakan topik-topik melalui judul dan abstrak artikel yang di- *input*kan kedalam *query* oleh *user* yang mengambil *dataset* jurnal terindeks GARUDA.
2. Mengetahui hasil rekomendasi *query* dengan jurnal berdasarkan judul dan abstrak yang di*input* kan menggunakan metode KNN.

1.3 Pembatasan Masalah

Batasan masalah dari penulisan tugas akhir ini adalah sebagai berikut:

1. Hanya menggunakan 3 *subject* jurnal yaitu *Computer Science* & IT, *Economics, Econometrics* & *Finance,* serta *Public Health* dengan total *abstract* dan *title* artikel- sejumlah 2.766, dan jurnal yang berjumlah 28.
2. *Dataset* yang digunakan hanya menggunakan jurnal yang menggunakan bahasa Indonesia dan bersumber dari jurnal terindeks GARUDA.

1.4 Tujuan Tugas Akhir

Tujuan dari Tugas Akhir ini adalah untuk mengimplementasikan metode LDA dan metode KNN dalam mencari jurnal yang relevan dengan *abstract* dan judul artikel yang telah dipublikasikan di GARUDA.

1.5 Manfaat

Manfaat dari Tugas Akhir ini adalah sebagai berikut:

1. Mempermudah para peneliti dalam mencari artikel yang relevan dengan jurnal yang ada pada platform GARUDA.
2. Menggabungkan LDA dan KNN dapat meningkatkan kualitas, relevansi, dan personalisasi rekomendasi yang diberikan kepada pengguna.

1.6 Sistematika Penulisan

Sistematika penulisan yang akan digunakan oleh penulis dalam sebuah pembuatan laporan tugas akhir ini adalah sebagai berikut:

BAB I : PENDAHULUAN

Pada Bab 1, penulis mengutarakan urgensi dari penelitian yang diangkat, mulai dari penulisan latar belakang, membuat rumusan masalah, membatasi permasalahan yang dibahas, serta tujuan dan manfaat yang diperoleh, dan sistematika penulisan.

BAB II : TINJAUAN PUSTAKA DAN DASAR TEORI

Pada bab 2, penulis memuat dasar teori yang digunakan, serta rujukan dari penelitian terdahulu yang akan digunakan dalam perancangan sistem, dan membantu penulis untuk memahami teori yang berhubungan dengan *Topic Modelling*, *Latent Dirichlet Allocation* (LDA), dan *K-Nearest Nighbor* (KNN) selama proses penelitian.

BAB III : METODE PENELITIAN

Pada bab 3, penulis mengungkapkan proses dan tahapan penelitian yang dimulai dari mendapatkan dataset hingga proses pemodelan topik data yang ada.

BAB IV : HASIL DAN ANALISIS PENELITIAN

Pada bab 4, penulis mengungkapkan hasil penelitian, dimulai dengan hasil akhir sistem, klasifikasi data uji, dan akurasi dari sistem.

BAB V : KESIMPULAN DAN SARAN

Pada bab 5, penulis memaparkan kesimpulan dari hasil proses penelitian mulai dari awal sampai akhir.

BAB II
TINJAUAN PUSTAKA DAN DASAR TEORI

2.1 Tinjauan Pustaka

Sistem rekomendasi adalah sebuah aplikasi yang berfungsi untuk memprediksi suatu item yang menarik bagi *user*, dan terbagi menjadi dua tipe metode yang diterapkan pada sistem rekomendasi tersebut, yaitu *collaborative filtering* dan *content based filtering* (Badriyah *dkk.,* 2018). Dalam hal ini, pembuatan sistem rekomendasi jurnal dimaksudkan untuk mendapatkan kesamaan topik yang relevan dengan artikel yang sedang dicari dan terhubung dengan *abstract* dan judul jurnal.

Menurut Soyusiawaty dkk., (2018), sistem rekomendasi memiliki keterkaitan dengan *Information Retrieval* (IR) yakni merupakan sub-bidang yang berkaitan dengan representasi, penyimpanan, dan akses informasi. Hal ini termasuk mengambil informasi dari koleksi *database* yang mencakup proses menampilkan, menyimpan, dan mencari kumpulan data untuk menghasilkan pengetahuan sebagai tanggapan terhadap permintaan pengguna yang biasa disebut *query*.

Dalam salah satu penelitian oleh Bhat dkk., (2020) yang mengembangkan pemodelan topik *Latent Dirichlet Allocation* (LDA) dengan *Deep neural networking* yang disebut sebagai DeepLDA yang menggunakan dua model *Deep Neural Network* (2NN DeepLDA dan 3NN DeepLDA). Hasil yang diperoleh dari LDA kemudian dibandingkan dengan model (2NNDeepLDA dan 3NNDeepLDA) menggunakan metode *Support Vector Machine* (SVM). Sehingga hasil model yang diusulkan menunjukkan akurasi yang signifikan dengan sedikit usaha komputasi dibandingkan dengan metode LDA.

Penelitian lain yang dilakukan oleh Sanjaya, (2021) mengenai implementasi LDA, dengan perwakilan dokumen didasarkan pada topik yang diproduksi oleh LDA, yang penelitian awalnya menghitung nilai kesamaan kosin antara pasangan dokumen. Pada saat itu, 3000 kata digunakan untuk mewakili dokumen, dan ini menghasilkan tingkat akurasi 62% untuk mengklasifikasikan dokumen secara

otomatis dalam bahasa Bali.

Disamping itu, penelitian yang dilakukan oleh Badieah dkk., (2023) yang mengusulkan model rekomendasi tempat publikasi berbasis konten menggunakan jurnal terindeks GARUDA, dengan menggunakan kombinasi judul dan abstrak makalah yang digunakan sebagai masukan *query* untuk rekomendasi. Kemudian direpresentasikan sebagai vektor menggunakan metode TF-IDF (*Term Frequency-Inverse Document Frequency*), dan kemiripan artikel yang diukur dengan *cosine similarity* menggunakan vektor dari *query* dan dataset artikel jurnal.

Berdasarkan penelitian yang telah dilakukan sebelumnya, yaitu memodelkan rekomendasi tempat publikasi dengan menggunakan metode TF-IDF yang melihat kemiripan artikel dengan *cosine similarity*, sehingga pada penelitian ini akan dibuat sistem rekomendasi jurnal menggunakan dengan menggabungkan metode *Latent Dirichlet Allocation* (LDA) dalam implementasi pemodelan topik yang juga menggunakan KNN sebagai acuan dalam melihat pendekatan pada setiap topik yang diambil dari abstrak dan judul dari artikel-artikel yang terkandung didalam masing-masing jurnal.

2.2 Dasar Teori

2.2.1 Sistem Rekomendasi

Sistem rekomendasi merupakan sistem penyaringan informasi yang menggunakan preferensi orang lain untuk membantu pengguna menemukan konten, barang, atau layanan seperti situs web, buku, produk digital, film, lagu, wisata, dan materi pendidikan (Khoury, 2018). Desain sistem rekomendasi menjadi tahap perantara yang penting dalam memetakan spesifikasi atau kebutuhan aplikasi yang akan dibangun. Fokus utama dari sistem ini adalah memberikan pengalaman yang dipersonalisasi bagi pengguna, di mana dalam kasus ini, pengguna adalah para peneliti. Berdasarkan hal tersebut, menjadi kunci dalam memahami kebutuhan dan preferensi pengguna, yang kemudian digunakan untuk menyusun rekomendasi yang lebih relevan. Sistem ini memanfaatkan informasi seperti preferensi masa lalu, riwayat pencarian, dan data terkait lainnya untuk menghasilkan rekomendasi yang sesuai. Bagi peneliti yang mungkin belum memiliki banyak pengalaman atau pengetahuan dalam memilih dari berbagai

alternatif, sistem rekomendasi menjadi alat yang sangat berguna. Dengan menyediakan rekomendasi yang dipersonalisasi, sistem ini membantu peneliti dalam mengevaluasi alternatif yang lebih relevan dan sesuai dengan kebutuhan.

Secara umum, terdapat dua jenis utama dari sistem rekomendasi, yaitu sistem yang dipersonalisasi dan yang tidak dipersonalisasi. Namun, penelitian lebih cenderung memusatkan perhatian pada pengembangan sistem rekomendasi yang dapat memberikan pengalaman yang lebih pribadi dan relevan bagi pengguna, seiring dengan meningkatnya permintaan untuk rekomendasi yang disesuaikan dengan preferensi individual. Dengan demikian, pengembangan sistem rekomendasi yang lebih canggih dan dipersonalisasi terus menjadi fokus dalam dunia penelitian (Munawar *dkk.*, 2021).

Tujuan dari sistem rekomendasi tidak semata-mata terbatas pada pembentukan rekomendasi, prediksi peringkat, atau penyusunan peringkat item. Inti dari tujuan sejatinya adalah untuk memberikan bantuan dan dukungan kepada pengguna. Lebih dari sekadar menghasilkan rekomendasi, sistem ini bertujuan untuk menjadi alat yang membantu pengguna dalam mengatasi kebingungan atau kebingungan yang mungkin mereka alami saat mencari informasi atau objek yang sesuai. Dengan demikian, fokusnya adalah pada pemberdayaan pengguna melalui layanan yang tidak hanya mengandalkan prediksi atau peringkat semata, tetapi juga memberikan nilai tambah dalam bentuk bantuan langsung pada pihak pengguna atau peneliti.

Dalam perspektif ini, penting untuk mempertimbangkan peran sistem rekomendasi sebagai alat pendukung yang dapat meningkatkan pengalaman pengguna yaitu peneliti secara keseluruhan. Oleh karena itu, mengembangkan sistem rekomendasi yang tidak hanya efektif dalam memberikan rekomendasi yang akurat tetapi juga secara proaktif berkontribusi pada pemahaman dan kepuasan pengguna menjadi suatu prioritas (Munawar *dkk.*, 2021). Rekomendasi yang disediakan oleh sistem dapat menjadi landasan yang berguna dalam mendukung pengambilan keputusan oleh pengguna. Dengan mempertimbangkan saran-saran yang diberikan, pengguna dapat memiliki panduan yang lebih baik dalam menentukan pilihan mereka. Sistem rekomendasi membantu

menyederhanakan kompleksitas proses pengambilan keputusan dengan menyajikan opsi atau item yang sesuai dengan preferensi atau kebutuhan pengguna. sistem rekomendasi tidak hanya memfasilitasi penemuan item yang sesuai tetapi juga memberikan kontribusi positif terhadap kemudahan dan kepercayaan dalam pengambilan keputusan pengguna. Dalam hal ini sistem rekomendasi merujuk pada jurnal, dengan adanya sistem rekomendasi untuk jurnal dapat memberikan rekomendasi jurnal yang relevan dengan *abstract* dan judul yang dimasukkan kedalam *query*.

Sistem rekomendasi memberikan nilai tambah dalam membantu pengguna menavigasi melalui berbagai pilihan yang mungkin ada, meningkatkan kenyamanan, dan menyederhanakan proses pengambilan keputusan (Larasati *dkk.*, 2021).

2.2.2 *Topic Modelling*

Topic Modelling adalah jenis model statistik yang tidak memiliki pelabelan teks dengan tujuan untuk menemukan topik atau tema secara garis besar dari sekumpulan korpus (Mouhoub dan Al Helal, 2018). Selain itu, pemodelan topik merupakan algoritma yang mengembangkan cara baru untuk mencari, menelusuri, dan meringkas arsip teks berukuran besar dalam pencarian informasi termasuk saran tag, klasifikasi teks, ekstraksi kata kunci, dan pencarian kesamaan. *Topic Modelling* menjadi alat analisis yang berguna untuk menggambarkan esensi dari sejumlah besar teks dan mempermudah pemahaman tentang pokok-pokok bahasan atau tema yang ada di dalamnya.

Topic Modelling adalah suatu pendekatan analisis teks yang bertujuan untuk mengidentifikasi dan memahami pola tertentu yang muncul dari sebuah kumpulan teks. Tujuan utama dari *topic modelling* adalah memberikan gambaran umum atau ringkasan tentang topik utama yang dibahas dalam teks tersebut. Dengan menggunakan teknik pemodelan topik, sistem dapat mengelompokkan kata-kata atau frasa-frasa dalam teks ke dalam topik atau ide tertentu berdasarkan kemiripan makna atau konteks. Hal ini membantu dalam mengidentifikasi pola tematik atau fokus utama dari teks tersebut. Sebagai contoh, teknik pemodelan topik dapat digunakan untuk mengidentifikasi topik utama dari sejumlah artikel berita atau

dokumen ilmiah, memungkinkan pengguna untuk dengan cepat memahami isu-isu atau informasi kunci yang dibahas (Cahyono *dkk.*, 2023).

Topic Modelling adalah suatu pendekatan analisis data teks yang fokus pada pengelompokan dokumen berdasarkan topik tertentu. Hal ini melibatkan proses clustering yang mengelompokkan dokumen berdasarkan kemiripannya, sehingga dokumen-dokumen yang memiliki topik serupa ditempatkan bersama. Meskipun tujuan *topic modeling* sejalan dengan klasifikasi, namun prosedurnya berbeda. Tujuan utama dari *topic modeling* adalah memperoleh model topik abstrak pada kumpulan dokumen. Ini melibatkan memberikan tinjauan pada dokumen-dokumen berdasarkan topik tersebut, dan kemudian menggunakan tinjauan tersebut untuk mengelompokkan dokumen. Dengan kata lain, *topic modeling* bertujuan untuk mengungkap struktur topik yang mendasari dalam suatu susunan teks.

Selain itu, tujuan *topic modeling* mencakup penemuan bahwa setiap topik dapat dibentuk dari sejumlah kata tertentu yang membentuk subjek. Beberapa subjek dengan probabilitas yang sesuai didalam dokumen akan menghasilkan model interpretasi konten teks yang lebih rumit. Daftar subjek yang seringkali muncul dalam dokumen berdasarkan pola tertentu merupakan hasil pemodelan topik, yang mana akan membantu pembaca memahami subjek utama yang dibahas dalam dokumen dengan menawarkan pemahaman yang lebih baik tentang organisasi dan substansi kumpulan makalah. Ini memberikan wawasan yang lebih dalam terhadap struktur dan isi dari kumpulan dokumen, memungkinkan pengguna untuk memahami topik utama yang dibahas dalam teks (Sari *dkk.*, 2022).

Topic modeling yaitu suatu teknik dalam *domain text mining* yang digunakan untuk menemukan pola kata dalam koleksi dokumen dengan memanfaatkan model probabilistik hirarki. Teknik ini memberikan kemudahan dalam menganalisis dokumen teks yang jumlahnya banyak dan belum terkategorisasi. Algoritma yang digunakan dalam *topic modeling* memiliki kemampuan untuk menemukan pola atau semantik tersembunyi yang terdapat dalam kumpulan dokumen dan mengelompokkannya ke dalam topik-topik

tertentu. Teknik ini membantu pengguna untuk secara efektif mengeksplorasi dan mengelompokkan dokumen berdasarkan topik-topik yang terkandung di dalamnya (Dwi dan Maulana, 2021).

1. *Latent Semantic Analysis* (LSA): LSA bekerja dengan mereduksi dimensi ruang kata-kata dalam dokumen dan mengidentifikasi hubungan semantik antar kata. Hal ini membantu dalam menemukan pola tersembunyi dan topik dalam teks.
2. *Probabilistic Latent Semantic Analysis* (PLSA): PLSA menggunakan model probabilistik untuk menentukan hubungan antara dokumen, kata, dan topik. Ini berfokus pada kemungkinan kemunculan kata-kata dalam dokumen tertentu dan membentuk topik.
3. *Latent Dirichlet Allocation* (LDA): LDA merupakan sebuah algoritma *topic modeling* yang dikenal banyak orang. *Topic modelling* jenis ini mengartikan dalam setiap dokumen merupakan campuran yang berasal pada topik-topik yang bermacam-macam, lalu tiap kata dalam dokumen berasal dari salah satu topik tersebut. LDA membantu mengidentifikasi distribusi topik di dalam korpus dokumen.

2.2.3 *Latent Dirichlet Allocation* (LDA)

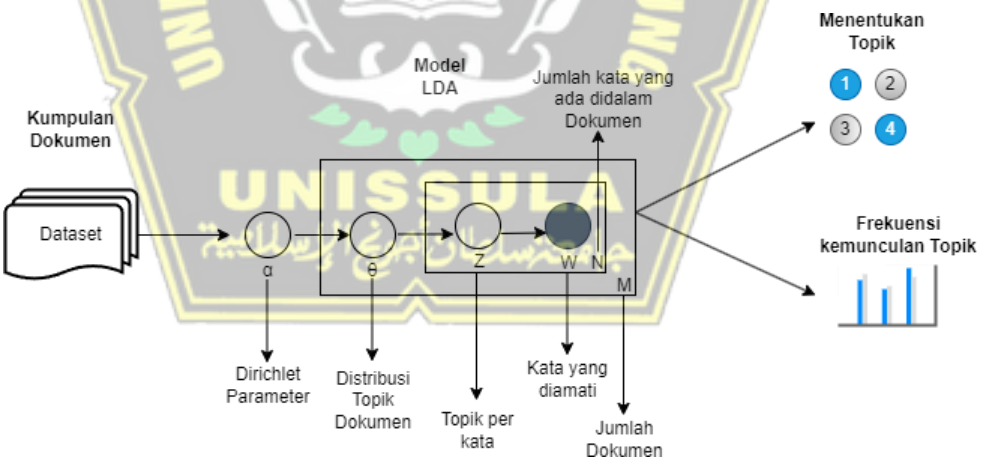
LDA memiliki definisi yang diartikan bahwa topik bermacam-macam yang terkandung dalam sebuah dokumen yang mempunyai probabilitas tertentu. Algoritma yang ada didalam LDA mampu untuk mendeteksi topik yang ada pada suatu koleksi dokumen beserta besarnya kemunculan topik tersebut pada koleksi dokumen maupun di dokumen tertentu (Kurniawan *dkk.*, 2022). *Latent Dirichlet Allocation* (LDA) merupakan metode yang populer dalam ekstraksi aspek atau topik, terutama dalam pendekatan *unsupervised learning*.

LDA yaitu sebuah algoritma dengan penggunanya banyak orang dalam hal *topic modelling*, dimana tujuannya adalah untuk mengidentifikasi topik atau aspek yang mendasari dalam suatu koleksi dokumen tanpa adanya anotasi atau label topik. LDA berusaha mengidentifikasi distribusi topik yang terkandung pada dokumen untuk mengidentifikasi kata per kata mana yang dapat menggambarkan setiap topik. LDA efektif mengkalkulasi rekomendasi topik ulasan teratas Ini

dilakukan secara probabilistik, di mana setiap dokumen dianggap sebagai campuran dari beberapa topik, dan setiap kata dalam dokumen berasal dari salah satu topik tersebut. Oleh karena itu, LDA telah menjadi pilihan utama dalam analisis teks untuk ekstraksi topik secara otomatis, terutama ketika datanya bersifat tidak terstruktur dan tidak diberi label (Adimanggala *dkk.*, 2021).

Latent Dirichlet Allocation (LDA) yakni sebuah teknik yang digunakan guna mengetahui topik-topik yang terkandung dalam suatu koleksi data serta untuk menentukan perbandingan kemunculan setiap topik tersebut. LDA adalah salah satu teknik dalam analisis topik yang bekerja secara probabilistik untuk mengidentifikasi distribusi topik. Cara kerja LDA melibatkan asumsi bahwa setiap dokumen merupakan campuran yang berasal dari banyaknya topik, dan setiap kata dalam dokumen berasal dari salah satu topik tersebut. Dengan menggunakan model probabilitas, LDA mencoba mengidentifikasi distribusi topik secara tersembunyi dalam kumpulan dokumen (Sari *dkk.*, 2022).

Berikut gambaran Diagram mengenai proses *Latent Dirichlet Allocation* (LDA).



Gambar 2. 1 Diagram Skema LDA

Latent Dirichlet Allocation (LDA), yang dimodelkan berdasarkan distribusi dan proses *Dirichlet*, yaitu mengkategorikan atau membagi teks menjadi dokumen dan kata berdasarkan topik. Langkah pertamanya adalah memulai dengan parameter yang mengambil jumlah dokumen, kata, topik, iterasi, serta koefisien LDA. Dalam LDA tentukan jumlah topiknya, sehingga ada satu topik yang mengandung lebih dari satu topik, apabila jumlah topik lebih besar dari jumlah topik yang

sebenarnya. Hal ini karena algoritma LDA bergantung pada distribusi kata per kata pada dokumen untuk menyelidiki benarkah kata-kata yang muncul berasal dari bahasa yang sama.

Langkah berikutnya yaitu memberi tanda suatu kata menggunakan topik yang sudah ditentukan secara acak berdasarkan distribusi *dirichlet*nya. Semua topik ada dalam dokumen tersebut. Selanjutnya, tahap iterasi dimana parameter dapat dibuat untuk mengidentifikasi persebaran jumlah topik dalam dokumen dan persebaran kata dari topik tersebut. Parameter ini terus dievaluasi untuk mengetahui seberapa umum kata terhadap topik dan seberapa umum topik tersebut berada dalam dokumen. Setelah berulang kali dilakukan dan mencapai hasil akhir dimana, semua kumpulan kata- kata yang memiliki arti sama digabungkan ke dalam satu topik.

2.2.4 K-*Nearest Neighbors* (KNN)

K-*Nearest Neighbors* (KNN) yaitu sebuah teknik pembelajaran mesin yang mudah namun produktif, ini efisien guna mengklasifikasi dan regresi. Namun, ini lebih umum dipergunakan dalam memprediksi dan mengklasifikasi. KNN bekerja dengan mengelompokkan data ke dalam kluster atau subset yang koheren dan mengklasifikasikan data yang di*input* dapat dilihat melalui kemiripannya berdasarkan data yang sebelum di*training*. Masukan tersebut ditugaskan ke kelas yang paling banyak berbagi tetangga terdekatnya. Hal ini membuat KNN dipergunakan untuk dapat mengklasifikasikan data berdasarkan contoh pelatihan terdekat atau tetangga di wilayah tertentu (Taunk *dkk*., 2019).

KNN adalah teknik klasifikasi berdasarkan seberapa jauh beberapa set data historis dari data yang baru dimasukkan. Dengan membandingkan beban beberapa atribut yang ada, teknik K-*Nearest Neighbors*, yang digunakan dalam pencarian kasus, menentukan seberapa dekat instansi baru dengan kasus sebelumnya. Teknik K-*Nearest Neighbors* bertujuan untuk menentukan pemisahan antara dua set data. Pengukuran seberapa dekat atau sebanding set data pengujian dan pelatihan satu sama lain dapat ditentukan menggunakan nilai jarak yang dihitung. Nilai K pada *Nearest Neighbors* berarti K-data terdekat dari data uji (Sebastian, 2019).

Algoritma pengajaran yang diawasi untuk KNN mengklasifikasikan

instance (jarak) yang baru berdasarkan mayoritas kategori pada KNN. Kelas yang paling banyak muncul akan menjadi kelas hasil klasifikasi. Tujuan algoritma ini adalah mengklasifikasikan objek baru berdasarkan atribut dan sampel pelatihan (Abadi, 2022). *K-Nearest Neighbors* (KNN) yakni teknik yang seringkali digunakan dalam mengklasifikasikan data. Teknik ini dipakai untuk mengklasifikasikan objek dari data pembelajaran yang memiliki jarak tetangga paling dekat atau memiliki nilai selisih yang kecil dengan objek tersebut (Koda *dkk.,* 2020).

Algoritma K-*Nearest Neighbors* (KNN) menjadi teknik umum yang dipergunakan dalam melakukan klasifikasi pada objek yang berdasarkan data *training* yang memiliki jarak paling terdekat dengan objek yang baru. Langkah awal pada penerapan KNN ialah menentukan nilai K, yang mewakili jumlah tetangga yang akan dipertimbangkan dalam proses klasifikasi. Setelah nilai K ditentukan, langkah berikutnya adalah mengukur jarak antara objek baru (*data testing*) dengan tetangga terdekatnya dalam data pembelajaran (*data training*). Jarak ini dapat dihitung menggunakan berbagai metrik, seperti *Euclidean distance* atau *manhattan distance*. Dalam penelitian ini, KNN dihitung dengan jarak *minkowski* yang merupakan jarak didalam ruang vektor dengan norma diartikan dan dianggap sebagai persamaan antara jarak *euclidean* dan *manhattan* (Manullang *dkk.*, 2021).

Berikut adalah rumus KNN dengan *metric Minkowski Distance*:

$$d(x,y) = (\sum_{i=1}^{n} |x_i - y_i|^p)^{\frac{1}{p}} \quad (1)$$

Keterangan :

$d(x, y)$ = Jarak antara x dan y

xi = Nilai dari objek x pada variabel ke-i

yi = Nilai dari objek y pada variabel ke-i

n = Jumlah variabel yang diamati

p = Power

Penting untuk menekankan bahwa pemilihan nilai K dapat mempengaruhi hasil klasifikasi. Nilai K yang terkecil dapat menghasilkan model menjadi sensitif terhadap *noise*, sedangkan nilai K yang terbesar dapat membuat model menghasilkan yang terlalu umum. Oleh karena itu, pemilihan K yang tepat adalah salah satu aspek kritis dalam penerapan KNN. Setelah jarak diukur, objek baru akan diklasifikasikan berdasarkan mayoritas kelas dari tetangga terdekatnya. Melalui implementasi algoritma KNN, maka dapat dicapai hasil klasifikasi yang cukup baik tanpa harus membuat asumsi yang kompleks tentang distribusi data. Meskipun sederhana, KNN tetap menjadi pilihan yang efektif terutama dalam kasus- kasus di mana hubungan antara fitur dan kelas tidak dapat dijelaskan dengan fungsi matematis yang terdefinisi dengan baik.

2.2.5 Garba Rujukan Digital (GARUDA)

Dalam dunia akademik, Portal Garuda adalah platform yang membantu untuk mengakses dan menemukan artikel dalam jurnal akademik, repositori, arsip, dan berbagai koleksi karya ilmiah (Wijaya, 2022). *Platform* pengumpulan referensi ilmiah yang berasal dari Indonesia, yakni laman akses untuk semua karya ilmiah yang dibuat oleh para akademisi dan peneliti. Didalam Garuda berisi *e-journal*, artikel, tugas akhir, karya ilmiah, serta laporan penelitian yang mana dapat dikembangkan oleh Direktorat P2M-Dikti Kemdiknas RI. Pada saat ini jumlah artikel trindeks Garuda berjumlah 2.966.633 artikel dari 20.648 jurnal dengan 3.824 penerbit, dan 305 *conferences* yang terdiri dari 40 subjek.

KemenRistek (Kementrian Riset, Teknologi, dan Pendidikan Tinggi Republik Indonesia) meluncurkan Garuda Kemendikbud (Kementrian Pendidikan dan Kebudayaan) Dikti pertama pada tahun 2018. Sebelumnya, sistem ini dikenal sebagai Indonesia *Publication Index* (IPI). Karena karya ilmiah mereka semakin berkualitas dan penelitian mereka tidak duplikat, peneliti, ilmuwan, dan akademisi perguruan tinggi akan memperoleh keuntungan dari keberadaan portal Garuda.

Portal ini memungkinkan untuk para peneliti melihat perkembangan ilmu pengetahuan dan kemampuan para ahlinya di Indonesia (Sa'adah, 2022). Disamping itu, Institusi dan lembaga kontributor juga memperoleh keuntungan yang signifikan dalam mendorong koleksinya, seperti meningkatkan akses, sitasi,

perangkingan, dan lainnya. Hal ini pula yang menjadi tujuan adanya platform Garuda yakni, meningkatkan ketersediaan dan keterjangkauan informasi umum dan ilmiah yang dapat diakses dengan cepat, membangun jaringan perpustakaan digital di Indonesia.

Platform Garuda adalah sistem sumber informasi publikasi ilmiah di Indonesia dan media informasi pendataan dan pengukuran kinerja Iptek, yang mencakup jurnal, publikasi penelitian, penulis (dosen atau peneliti), dan afiliasi (instansi Iptek) yang dikelola oleh Kemendikbud. Platform ini menyediakan akses ke semua informasi tentang publikasi lokal, terutama jurnal berstandar internasional. Dengan informasi publikasi yang tersedia di platform tersebut, civitas akademika dapat menggunakannya sebagai referensi saat menyusun dan menjalankan penelitian, mengerjakan tugas akhir perkuliahan, dan tugas lainnya (Nasrullah, 2023). Adanya portal sumber informasi tersebut, menjadi sumber referensi yang lebih luas akan tersedia, yang tidak hanya berasal dari buku tetapi juga dari artikel jurnal yang paling baru yang mengikuti perkembangan ilmu terbaru.

2.2.6 *Countvectorizer*

Countvectorizer merupakan salah satu metode ekstraksi fitur teks yang umum digunakan dalam pemrosesan bahasa alami. Fungsinya adalah mengubah teks kata-kata menjadi matriks frekuensi kata, dimana setiap baris mewakili satu dokumen atau kalimat, dan setiap kolom mewakili kata tertentu. Metode *fit_transform* dari *Countvectorizer* digunakan untuk menghitung jumlah kemunculan setiap kata dalam teks dan membangun vektor fitur. Tujuan utama dari *Countvectorizer* adalah untuk memperoleh setiap kata unik dari seluruh teks dan membangun kosakata (*vocabulary*) dari kata-kata tersebut. Kosakata ini nantinya akan digunakan untuk membentuk vektor fitur dari setiap dokumen atau kalimat. Proses ini memungkinkan representasi numerik dari teks, di mana setiap elemen vektor mencerminkan jumlah kemunculan kata tertentu dalam dokumen (Rahayu *dkk.*, 2023).

Melalui *Countvectorizer*, data teks dapat mengonversi menjadi representasi numerik yang dapat digunakan sebagai input untuk model pembelajaran mesin.

Teknik ini sering digunakan dalam klasifikasi teks, analisis sentimen, dan tugas pemrosesan bahasa. *Countvectorizer* adalah alat yang digunakan untuk menghitung frekuensi kemunculan kata-kata dalam suatu teks dan membentuk matriks yang merepresentasikan dataset dengan kata-kata sebagai fitur. Proses ini dimulai dengan merinci kalimat-kalimat dalam dokumen menjadi kata- kata atau token. *Countvectorizer* menciptakan matriks frekuensi kata-kata dengan membangun vektor untuk setiap dokumen. Vektor ini memiliki panjang yang sama dengan jumlah kata unik dalam keseluruhan kumpulan dokumen. Setiap elemen vektor menunjukkan berapa kali kata tertentu muncul dalam dokumen tersebut. Selain menghitung frekuensi kata-kata dalam setiap dokumen, *Countvectorizer* juga berperan dalam membentuk kosakata dari semua kata unik di seluruh dataset (Rabbani *dkk.,* 2023).

Kosakata ini berfungsi sebagai *database* cadangan kata-kata yang mendasari pembentukan vektor fitur. Matriks yang dihasilkan oleh *Countvectorizer* menyediakan representasi numerik dari dataset, memungkinkan penggunaannya dalam berbagai analisis dan model pembelajaran mesin. *Countvectorizer*, dengan pendekatan ini, menjadi alat penting dalam pemrosesan bahasa alami dan analisis teks untuk berbagai tujuan, seperti klasifikasi teks, analisis sentimen, atau penggalian informasi. *Countvectorizer* fokus pada representasi numerik dari teks tanpa memperhitungkan kelas atau label yang terkait dengan setiap dokumen. *Countvectorizer* bekerja dengan cara menghitung frekuensi kemunculan kata-kata dalam dokumen dan membentuk matriks frekuensi kata-kata. Setiap dokumen direpresentasikan sebagai vektor numerik, di mana setiap elemen vektor menunjukkan berapa kali kata tertentu muncul dalam dokumen tersebut. Namun, informasi tentang label atau kelas dokumen tidak diikutsertakan dalam proses ini. Sementara *Countvectorizer* memberikan representasi numerik yang efektif dari teks untuk digunakan dalam berbagai tugas pemrosesan bahasa alami, untuk beberapa kasus, terutama yang melibatkan klasifikasi dokumen (Lestandy *dkk.*, 2021).

2.2.7 Evaluasi

Menurut Rodrigues, (2018) Evaluasi merupakan langkah-langkah yang digunakan untuk mengukur kinerja dan efektivitas model rekomendasi supaya berjalan sesuai dengan diagram atau alur yang telah dibuat. Dalam hal ini, rekomendasi berkaitan dengan mengembalikan urutan peringkat item, dimana setiap item mungkin relevan atau tidak relevan dengan item yang berada di urutan paling bawah dalam daftar, yang memiliki kemungkinan relevan yang lebih rendah. Dalam hal merekomendasikan jurnal, peneliti hanya mempunyai waktu yang terbatas. Sehingga, mereka harus menerima rekomendasi jurnal yang paling relevan terlebih dahulu, baru kemudian rekomendasi yang mungkin kurang relevan. Untuk evaluasinya sendiri menggunakan metode *Mean Average Precision*@K (MAP@K) untuk mengevaluasi model pemberi rekomendasi. Dimana dalam proses evaluasi ini berarti melakukan pemeringkatan (*scoring*) tertinggi sebagai rekomendasi teratas lalu diikuti dengan pemeringkatan dengan skor dibawahnya.

Tahapan evaluasi yang pertama dengan metode MAP@K adalah menghitung presisi terlebih dahulu berdasarkan variabel K (P@K) yang merupakan K teratas item Variabel K adalah K teratas pada item yang direkomendasikan. Sebelum menggunakan metrik ini, perlu dilakukan pengukuran awal dengan menggunakan presisi@k (P@K) dan rata-rata (*Average)* Presisi@K (AP@K). Presisi sebagai proporsi tempat relevan yang diambil dari serangkaian tempat yang direkomendasikan. Namun, tingkat presisi ini tidak mencerminkan peringkat rekomendasi tempat terkait. Untuk memasukkan peringkat ke dalam formula presisi, P@K digunakan dan didefinisikan sebagai proporsi tempat yang relevan di antara tempat-tempat yang direkomendasikan oleh K teratas. P@K dapat dirumuskan sebagai berikut.

$$P@K = \frac{\sum Relevant\ venues\ in\ top\ K\ recommendations}{\sum venues\ in\ top\ K\ recommendations} \quad (2)$$

P@K akan mencerminkan seberapa relevan tempat publikasi yang direkomendasikan oleh model di top K. Dalam tulisan ini, kami bereksperimen dengan berbagai rekomendasi Top K mulai dari Top 1 hingga Top

10. Oleh karena itu, presisi untuk semua Top K perlu diukur menggunakan AP@K. AP@K adalah jumlah dari presisi@k dimana K_{th} peringkat jurnal yang relevan (rel(k)) dibagi dengan jumlah total jurnal rekomendasi yang relevan (r) dalam rekomendasi K teratas. Rumus matematika AP@K adalah sebagai berikut :

$$AP@K = \frac{1}{r}\sum Precision@k.rel(k) \quad (3)$$

dimana:

$$rel(k) = \begin{cases} 0, & if\ venues\ at\ k_{th}\ rank\ is\ relevant \\ 1, & otherwise \end{cases}$$

Nilai dari rel (k) menunjukkan apakah urutan K tempat itu relevan. Jika relevan, nilainya 1, dan jika tidak, maka penalti adalah 0. Pada AP@K yang disebutkan sebelumnya diterapkan pada satu titik data, sedangkan dalam penelitian ini Peneliti memakai data *test* sejumlah 105 data. Kemudian menghitung MAP@K menggunakan rumus berikut.

$$MAP@K = \frac{1}{M}\sum_{j=0}^{M}\frac{1}{r}\sum_{k=1}^{k} Presicion@\text{k}.\text{rel(k)} \quad (4)$$

Pada rumus MAP@K diatas adalah menghitung rata-rata dari total presisi dari sistem rekomendasi yang telah dihitung, sehingga menghasilkan besaran rata-rata rekomendasi yang ditunjukkan.

BAB III
METODOLOGI PENELITIAN

3.1 Metode Penelitian

Penelitian ini menggunakan metode atau algoritma yang disebut *Latent Dirichlet Allocation* (LDA) dan *Cosine Similarity*. Dalam penelitian ini, Kedua pendekatan tersebut digabungkan untuk mencari sebagai sistem rekomendasi yang dapat memproses, mengklusterisasi, dan mengkategorikan informasi topik berdasarkan *abstract* dan judul yang dimasukkan. Langkah- langkah yang harus dilakukan dalam penelitian ini adalah sebagai berikut :

3.1.1 Studi Literature

Dalam penelitian ini akan dilakukan tinjauan terhadap beberapa *e-book*, makalah, jurnal, tesis, dan skripsi terdahulu akan diulas selain mengunjungi berbagai situs web. Tujuan dari tinjauan ini adalah untuk mempelajari teori di balik konsep sistem rekomendasi jurnal, *topic modelling,* dan metode *Latent Dirichlet Allocation* (LDA).

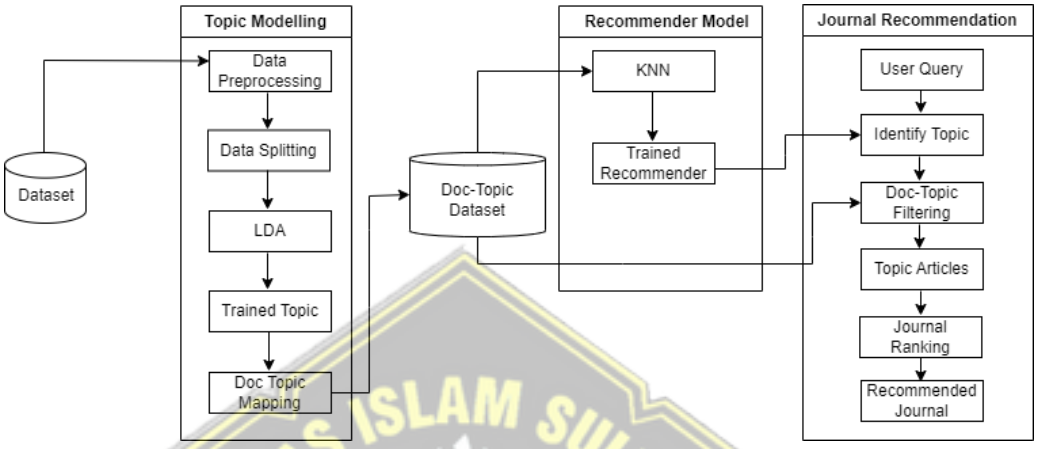
3.1.2 Pengumpulan Data

Pada tahap ini sebagai persiapan awal dan pengembangan model adalah dengan mengumpulkan data. Data yang digunakan adalah menggunakan dataset yang sama pada penelitian sebelumnya yang diambil dari platform GARUDA, perbedaannya adalah dalam penelitian ini mengambil sebanyak 28 jurnal, dan 3 *subject* jurnal, yang diantaranya adalah '*Computer Science & IT*', '*Economics, Econometrics & Finance*', dan '*Public Health*' dengan mengambil abstrak dan judul artikel yang menggunakan bahasa Indonesia, dengan jumlah sebanyak 2.766.

3.1.3 Metode Perancangan Arsitektur Model

Dalam tahap ini Peneliti merencanakan langkah-langkah proses perancangan arsitektur model. Dengan tahap-tahapannya mengacu pada gambar 7.2 ada 2 langkah yakni melakukan *data preprocessing*, LDA (*Latent Dirichlet Allocation*) untuk menemukan topik apa saja yang ada pada *dataset*, lalu dari dokumen-dokumen yang ada dilakukan pemtaan pada masing-masing topik

dengan menggunakan KNN. Setelah itu, dilakukan *training* data yang juga menggunakan KNN untuk selanjutnya didapatkan rekomendasi jurnal-jurnal terkait topik yang dimasukkan kedalam *query* oleh *user*.

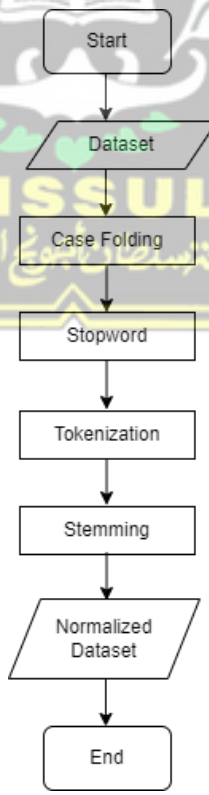


Gambar 3. 1 Perancangan Arsitektur Model

3.1.4 Tahapan Perancangan Model

1) *Topic Modelling*

a) Data *Preprocessing*



Gambar 3. 2 *Flowchart Preprocessing*

Tahap awal perancangan model adalah melakukan data *preprocessing* untuk mengolah *dataset* yang sudah siap digunakan, dengan mempersiapkan *dataset* yang menampilkan isi dari *dataset*, lalu melakukan *casefolding* yaitu mengubah huruf kapital yang ada didalam *dataset* menjadi huruf kecil seluruhnya. Selanjutnya adalah *tokenization* yaitu pemecahan teks menjadi bentuk kata per kata yang disebut sebagai token, melakukan *stopword removal* yaitu menghilangkan kata-kata umum yang sering muncul, tetapi tidak meberikan informasi yang penting, contohnya seperti, ‘di’, ‘dan’, ‘ke’, ‘itu’, ‘yang’, ‘juga’, terdapat pada token berdasarkan daftar *stopword*. Karena dataset menggunakan bahasa Indonesia, maka *stopword removal* menghilangkan kata tidak penting yang juga memakai bahasa Indonesia. Setelah itu, *stemming* untuk mengubah kata ke bentuk dasarnya, biasanya kata-kata tersebut memiliki imbuhan dan akhiran yang diantaranya ber-, di-, ter-, pen-, per-, se-, ke-. me-, -kan, -nya, -an. Lalu *normalized dataset* adalah pengaturan ulang agar lebih rapi dan bebas dari redudansi data, agar hasil *preprocessing* dapat digunakan dengan baik.

b) Data *Splitting*

Tahap ini dilakukan setelah proses data *preprocessing* pada *dataset* yang sudah berbentuk token untuk membagi data *test* dan data *train* dengan jumlah data *train* sejumlah 2.661 dan data *test* sejumlah 105. Data *train* digunakan untuk melatih model, sedangkan data *test* untuk mengevaluasi dan melatih model yang telah di *training* mampu memprediksi label yang belum dipelajari oleh model.

c) Implementasi *Latent Dirichlet Allocation* (LDA)

Pada tahap ini melakukan pemodelan LDA pada *dataset* yang telah di *pre-processing,* LDA memprosesnya dengan mengambil topik-topik tersembunyi didalam dataset untuk di *cluster* kedalam topik-topiknya. Dengan proses iterasi maksimal sebanyak 5 iterasi dengan menentukan jumlah topik yang akan diuji..

d) *Trained Topic*

Tahap ini merupakan hasil dari topik-topik yang telah dimodelkan dengan LDA dilakukan *training topic* berupa *build* untuk mengambil topik probabilistik dengan niliai >= 0.5 yang melibatkan model dan *vectorizer*.

e) *Doc-Topic Mapping*

Tahap ini adalah memetakan dokumen-dokumen kedalam masing-masing *doc_id*, *topic_id*, dan *topic_probabilistik* yang menghasilkan *dataset* baru bernama *Doc-Topic* yang telah dimodelkan dengan pemodelan topik.

2) *Recommender Model*

a) K-*Nearest Neighbors* (KNN)

Kemudian, di tahap ini adalah melakukan pelabelan pada *dataset Doc-Topic*, sehingga *dataset* tersebut sudah *supervised learning*. Pada KNN menggunakan *metric minkowski* dan n_*neighbors* sebanyak 20 untuk melihat *similarity* atau kedekatan topik pada suatu dokumen.

b) *Trained Recommender*

Pada tahap ini, *trained recommender* melatih dokumen-dokumen berisi artikel untuk melihat kedekatan topik atau *similarity*nya pada suatu dokumen yang diambil dari fungsi *get_docs* untuk mengakses dan menganalisis dokumen-dokumen yang telah diproses, sehingga dapat digunakan untuk berbagai keperluan analisis identifikasi topik.

3) *Journal Recommendation*

a) *User Query*

Tahapan *user* memasukkan *query* berupa topik dalam bentuk judul dan abstrak suatu artikel.

b) *Identify Topic*

Mengidentifikasi topik yang dimasukkan *user* termasuk kedalam jenis topik nomor berapa, dan termasuk kedalam *doc_id* yang mana.

c) *Doc-Topic Filtering*

Pada tahap ini dilakukan untuk mengelompokkan *topic probabilistik* tertinggi yang sesuai dengan dokumen-dokumen yang dipertahankan pada hasil *identify topic* atau dapat diambil dari dataset *doc topic*.

d) *Topic Articles*

Mencocokkan *topik artikel* dengan kategori jurnal-jurnal yang ada.

e) *Journal Ranking*

Proses ini dilakukan untuk mengurutkan jurnal yang diurutkan berdasarkan rata-

rata dari topik probabilistik, tanpa mempertimbangkan topik tertentu.

f) *Recommended Journal*

Pada tahap ini adalah hasil akhir berupa list rekomendasi jurnal yang relevan dengan judul dan abstrak yang di-*input*kan pada *query*.

3.2 Analisis Kebutuhan

Penelitian ini menganalisis *software* apa saja yang akan dibutuhkan dalam pengembangan aplikasi, sehingga mulai dari proses *input* hingga hasil akhir dapat berjalan sesuai dengan yang diharapkan. berikut adalah daftar perangkat lunak yang digunakan dalam pengembangan sistem:

1. Python 3.10.1

Python merupakan bahasa pemrograman tingkat tinggi yang fleksibel dengan fokus pada keterbacaan dan kesederhanaan. Python, yang dikembangkan oleh Guido van Rossum dan pertama kali tersedia pada tahun 1991, telah berkembang menjadi salah satu bahasa pemrograman yang paling disukai dan sering digunakan di seluruh dunia. Python versi 3.10.1 digunakan sebagai bahasa pemrograman untuk penelitian ini karena memiliki *library* yang lengkap dan bersifat *open source*.

2. *Google Colaboratory*

Google Colaboratory adalah sebuah aplikasi *dari* platform Google yang digunakan untuk mengembangkan dan menjalankan kode dalam lingkungan komputasi interaktif. dalam penelitian ini, *google colab* digunakan untuk menulis dan menjalankan kode program, serta membuat penyimpanan catatan atau dokumentasi.

3. *Library Pandas*

Library Pandas *python* merupakan library yang digunakan untuk memproses data, mulai pembersihan data, manipulasi data, hingga melakukan analisis data. Dalam penelitian ini library Pandas digunakan untuk membuat laporan data pada saat *testing* dalam *google colab* dan juga digunakan dalam pengolahan data menjadi data JSON yang nantinya akan ditampilkan dalam tampilan *website*.

4. *Library* NumPy

NumPy (*Numerical Python*) adalah *library* python yang digunakan untuk bekerja dengan *array* dan juga memiliki fungsi yang bekerja dalam domain aljabar *linier*, transformasi *Fourier*, dan matriks.

5. *Library Scikit-Learn*

Library scikit-learn, adalah library yang digunakan untuk pembelajaran mesin (*machine learning*) yang ditulis dalam bahasa pemrograman Python.

6. *Library* NLTK (*Natural Language Tool Kit*)

Natural Language Toolkit (NLTK) adalah *library* atau *toolkit* populer dalam bahasa pemrograman Python yang digunakan untuk pemrosesan *Natural Language Processing* (NLP). dalam penelitian ini, NLTK digunakan dalam proses *preprocessing* atau pengolahan teks diantaranya *casefolding, stopwords, tokenization, stemming,* dan *normalized text*.

7. *Library Ktrain*

Ktrain saat ini mencakup dukungan out-of-the-box untuk model pelatihan pada teks, visi, grafik, dan data tabel. Sebagai wrapper sederhana untuk TensorFlow Keras, Selain itu, *ktrain* cukup fleksibel untuk digunakan dengan model dan format data kustom. dirancang untuk membuat pembelajaran mendalam dan AI lebih dapat diakses dan lebih mudah untuk diterapkan untuk kedua pendatang baru dan praktisi berpengalaman. Dengan hanya beberapa baris kode, ktrain memungkinkan pemrosesan dengan mudah dan cepat.

8. *Visual Studio Code*

Visual Studio Code dipilih sebagai text editor pada pengembangan aplikasi dalam penelitian ini, *Visual Studio Code* dipilih dikarenakan mendukung banyak Bahasa pemrograman dan framework, multi-platform, performa yang sangat cepat, mempunyai banyak extensions yang dapat mempermudah proses pengembangan.

9. *Framework Streamlit*

Streamlit ialah sebuah *framework* berbasis *Python* yang bersifat open-source yang dirancang untuk mempermudah pembuatan aplikasi web interaktif di bidang sains data dan pembelajaran mesin (UII, 2022). Pada penelitian ini, *framework*

streamlit digunakan sebagai pengembangan aplikasi *website* dikarenakan core dari sistem yang akan dikembangkan sendiri berbahasa *python*.

3.3 Analisis Sistem

Dalam tahapan analisi sistem, dilakukan analisa guna membuat alur kerja sistem yang akan dibuat dengan memakai *flowchart*. Langkah pertama adalah pengguna atau *user* membuka aplikasi. Kemudian akan terlihat halaman sistem rekomendasi dengan sebuah *text area*. *User* bisa meng-*input* kan *query* pada *text area* dengan mengetikkan judul dan abstrak dari sebuah artikel. Setelah itu, *user* bisa mengklik tombol "Temukan Rekomendasi" guna memproses *query* tersebut. Selanjutnya. *query* akan diproses dengan mencocokkan judul dan abstrak yang di*input*kan dengan masing-masing topik yang telah dilatih menggunakan model KNN yang ada didalam sistem. *Flowchart* talur kerja sistem dapat diamati pada gambar 3.3.

Gambar 3. 3 *Flowchart* Sistem Rekomendasi Jurnal

Dapat dilihat pada gambar 3.3 bahwa alur sistem sistem rekomendasi tersebut mempunyai beberapa tahap yang diuraikan sebagai berikut:

a. Pertama pengguna masuk ke tampilan sistem rekomendasi jurnal. pada tampilan rekomendasi, pengguna akan melihat bagian *appbar*, *body* dan bagian

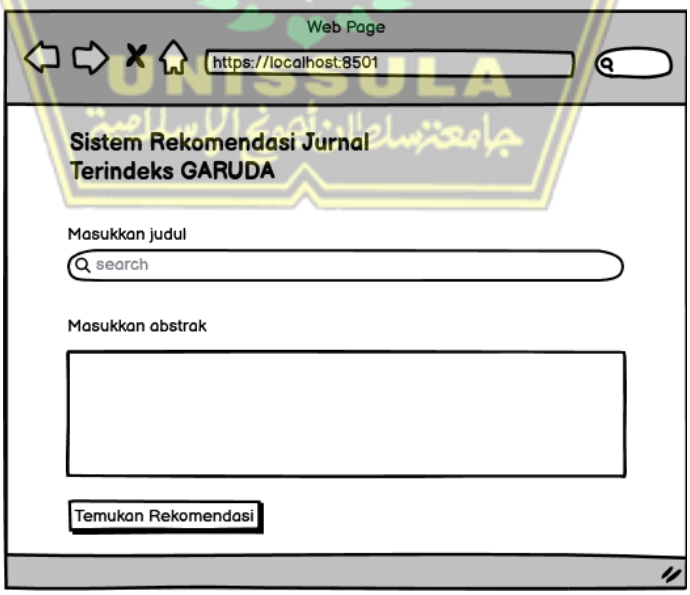
input text area.

b. Kemudian pengguna dapat menginputkan judul dan abstrak dari suatu artikel di bagian *input text area*, setelah pengguna meng*input*kan judul dan abstrak, pengguna dapat mengirimnya dengan cara menekan mengklik tombol atau *botton* bertuliskan "Temukan Rekomendasi" di bagian bawah *input query*.
c. Setelah *query* terkirim, maka sistem akan memproses judul dengan abstrak untuk dipetakan ke dalam *topic_id*, *doc_id*, *topic probabilistic*, dan jurnal yang cocok.
d. Jika teks judul dan abstrak yang dikirim memilik kecocokan dengan data yang sudah dilatih oleh model maka sistem akan memberikan hasil berupa rekomendasi jurnal yang sesuai berdasarkan tingginya nilai *topic probabilistic* dari judul dan abstrak artikel yang di*input*kan di bagian *input text area* atau *body*.

3.4 Perancangan *User Interface*

3.4.1 Halaman Awal Sistem Rekomendasi Jurnal

Halaman awal sistem rekomendasi jurnal adalah halaman tampilan sebelum memulai proses pencarian rekomendasi, dapat diamati seperti pada gambar 3.4.



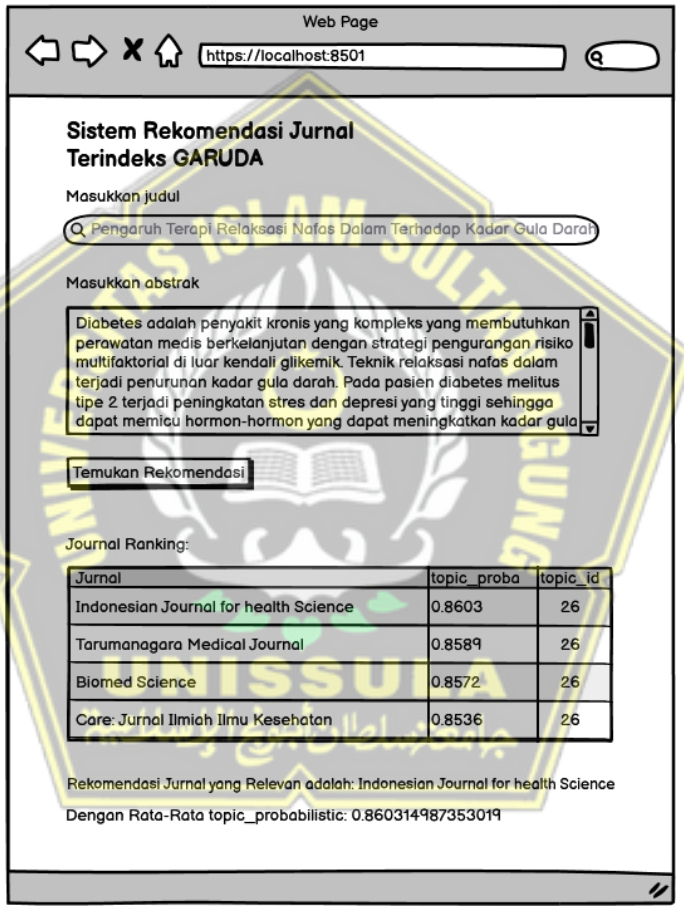
Gambar 3. 4 Halaman Awal Sistem Rekomendasi

Dapat dilihat pada gambar 3.4 bahwa tampilan awal sistem rekomendasi

jurnal yang diakses oleh *user* dapat melihat bagian *input text area* yang digunakan untuk mengetikkan judul dan abstrak suatu artikel. Lalu dibawah kotak *input text* area adalah tombol untuk pencarian rekomendasi.

3.4.2 Halaman Hasil Sistem Rekomendasi Jurnal

Pada halaman hasil rekomendasi jurnal yang dicari oleh *user* dengan mengetikkan *input* judul dan abstrak menampilkan hasil rekomendasi jurnalnya. Dapat dilihat pada gambar 3.5.



Gambar 3. 5 Halaman Rekomendasi

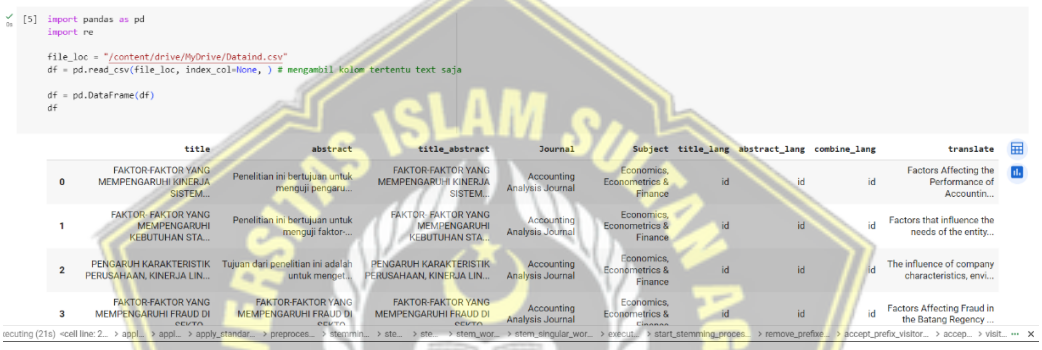
Terlihat pada gambar 3.5 yang menampilkan isi dari hasil proses rekomendasi yang dibutuhkan *user*. Dalam proses merekomendasikan hasil jurnal yang sesuai dengan *input* yang diberikan oleh *user*, sistem akan memberikan hasil rekomendasi berupa *journal ranking* berdasarkan *topic probabilistiv* tertinggi yang menunjukkan bahwa *input* yang diproses masuk kedalam jenis jurnal apa saja. Lalu dibawah tabel *journal ranking* ada hasil teratas jurnal yang paling

relevan yang dihitung berdasarkan rata-rata *topic probabilistic* nya. Penelitian ini akan mendemokan halaman web yang sudah dibuat kepada pengguna dan pengguna yang memakainya, apakah sudah merasa sesuai dengan apa yang diinginkan.

BAB IV
HASIL ANALISIS PENELITIAN

4.1 Hasil Penelitian

Data yang digunakan dalam bentuk *dataset* berisi kumpulan artikel sejumlah 2.766, dengan 28 jurnal, dan 3 jenis *subject* yang diambil dari *platform* jurnal terindeks GARUDA. Data tersebut telah dikelompokkan berdasarkan *title*, *abstract*, *title_abstract, journal*, *subject, title_lang, abstract_lang, combine_lang, translate* dalam bentuk file .csv. yang terlihat pada gambar 4.1



```
[5] import pandas as pd
    import re

    file_loc = "/content/drive/MyDrive/Dataind.csv"
    df = pd.read_csv(file_loc, index_col=None, ) # mengambil kolom tertentu text saja

    df = pd.DataFrame(df)
    df
```

	title	abstract	title_abstract	Journal	Subject	title_lang	abstract_lang	combine_lang	translate
0	FAKTOR-FAKTOR YANG MEMPENGARUHI KINERJA SISTEM...	Penelitian ini bertujuan untuk menguji pengaru...	FAKTOR-FAKTOR YANG MEMPENGARUHI KINERJA SISTEM...	Accounting Analysis Journal	Economics, Econometrics & Finance	id	id	id	Factors Affecting the Performance of Accountin...
1	FAKTOR- FAKTOR YANG MEMPENGARUHI KEBUTUHAN STA...	Penelitian ini bertujuan untuk menguji faktor-...	FAKTOR- FAKTOR YANG MEMPENGARUHI KEBUTUHAN STA...	Accounting Analysis Journal	Economics, Econometrics & Finance	id	id	id	Factors that influence the needs of the entity...
2	PENGARUH KARAKTERISTIK PERUSAHAAN, KINERJA LIN...	Tujuan dari penelitian ini adalah untuk menget...	PENGARUH KARAKTERISTIK PERUSAHAAN, KINERJA LIN...	Accounting Analysis Journal	Economics, Econometrics & Finance	id	id	id	The influence of company characteristics, envi...
3	FAKTOR-FAKTOR YANG MEMPENGARUHI FRAUD DI [illegible]	FAKTOR-FAKTOR YANG MEMPENGARUHI FRAUD DI [illegible]	FAKTOR-FAKTOR YANG MEMPENGARUHI FRAUD DI [illegible]	Accounting Analysis Journal	Economics, Econometrics & Finance	id	id	id	Factors Affecting Fraud in the Batang Regency ...

Gambar 4. 1 *Dataframe* dari *Dataset*

Pada gambar 4.1 adalah gambar tabel *dataframe* yang dihasilkan melalui *dataset* format .csv. Dari *dataframe* tersebut, dapat dilihat persebaran artikel mulai dari *title, abstract, title_abstract, Journal, Subject, title_lang, abstract_lang, combine_lang* sampai kolom paling kanan yaitu *translate*. Setelah melihat hasil *dataset* yang ditampilkan menjadi sebuah *dataframe*, diambil sejumkah tiga pada masing-masing kolom yang berisi data artikel yaitu kolom *title_abstract, journal*, dan *subject* untuk dilakukan tahapan pertama yaitu *preprocessing*. Tahapan pertama dalam melakukan *preprocessing* adalah dengan mengambil informasi yang terkandung didalam kolom *title_abstract* karena kolom tersebut merupakan bagian terpenting untuk dijadikan sebuah data yang dapat diolah dengan baik, seperti yang dapat dilihat pada tabel 4.1.

Tabel 4. 1 Hasil dari *Casefolding*

Sebelum	Sesudah
PENERAPAN WORD N-GRAM UNTUK SENTIMENT ANALYSIS REVIEW MENGGUNAKAN METODE SUPPORT VECTOR MACHINE (STUDI KASUS: APLIKASI SAMBARA) ABSTRACT Sambara application is an innovation from Bapenda West Java for motor vehicle tax services. The Sambara application expected can be provide efficiency, effectiveness and service improvement. The success of the application can be determined by conducting a sentiment review analysis. Sentiment analysis aims to detect polarity in the text in the form of negative or positive opinions, using text mining. At the text processing stage, the Word N-Gram feature is added as a word identification approach and for classification it uses the Support Vector Machine (SVM) method. This…	penerapan word ngram untuk sentiment analysis review menggunakan metode support vector machine studi kasus aplikasi sambara abstract sambara application is an innovation from bapenda west java for motor vehicle tax services the sambara application expected can be provide efficiency effectiveness and service improvement. the success of the application can be determined by conducting a sentiment review analysis sentiment analysis aims to detect polarity in the text in the form of negative or positive opinions using text mining at the text processing stage the word ngram feature is added as a word identification approach and for classification it uses the support vector machine svm method this…

Pada tabel 4.1, dilakukan proses *preprocessing* dengan mengubah huruf kapital menjadi huruf kecil (non-kapital) hanya pada kolom *title_abstract*, karena kolom tersebut yang nantinya akan dilatih dan diuji. Selain itu, dilakukan pula *remove special character* dan *remove punctuiation* untuk menghilangkan karakter khusus seperti tanda baca dan angka serta menghapus spasi berlebih pada data tersebut. Dalam penulisan kodingannya, menuliskan fungsi apa saja yang akan

digunakan kemudian memanggil metode *apply* pada kolom *title_abstract* untuk menerapkan fungsi tersebut disetiap kalimat yang ada dikolom tersebut.

Langkah selanjutnya yaitu menghilangkan setiap kata hubung dengan metode *stopword removal*, dalam hal ini menggunakan *statement* "*import string"* yaitu modul yang isinya adalah berbagai macam konstanta kata. Adapula *"import nltk"* atau *natural language toolkit* yang berisi berbagai macam fungsi dan *library* untuk tahap *preprocessing* ini, fungsi *stopword* untuk bahasa Indonesia yaitu dengan meng*input*kan bahasa 'indonesian' agar fungsi tersebut dapat mendefinisikan untuk menghilangkan kata yang tidak bermakna menggunakan bahasa Indonesia, kata yang tidak bermakna tersebut diantaranya adalah 'dan', 'ke, 'dari, 'untuk', 'yang'. Hasil dari proses *stopword removal* dapat dilihat pada tabel 4.2.

Tabel 4. 2 Hasil Proses *Stopword Removal*

Sebelum	**Sesudah**
penerapan word ngram untuk sentiment analysis review menggunakan metode support vector machine studi kasus aplikasi sambara abstract sambara application is an innovation from bapenda west java for motor vehicle tax services the sambara application expected can be provide efficiency effectiveness and service improvement the success of the application can be determined by conducting a sentiment review analysis sentiment analysis aims to detect polarity in the text in the form of negative or positive opinions using text mining at the text processing stage the word ngram	penerapan word ngram sentiment analysis review metode support vector machine studi aplikasi sambaraabstractsambara application is an innovation from bapenda west java for motor vehicle tax services the sambara application expected can be provide efficiency effectiveness and service improvement the success of the application can be determined by conducting a sentiment review analysis sentiment analysis aims to detect polarity in the text in the form of negative or positive opinions using text mining at the text processing stage the word ngram feature is added as a word identification approach and

feature is added as a word identification approach and for classification it uses the support vector machine svm method this…	for classification it uses the support vector machine svm method this study aims to determine the application of word ngram the results of the accuracy value using the svm method this…

Dapat dilihat pada tabel 4.2 bahwa proses *stopword* dalam menghilangkan kata Fungsi proses *stopword* ini juga agar pemodelan topik yang dihasilkan nanti tidak hanya berisi kata yang tidak bermakna, yang membuat pemodelan topik kurang maksimal. Setelah menghilangkan kata hubung yang terkandung didalam *dataset*, beralih ke proses selanjutnya yaitu tokenisasi yang dapat dilihat pada tabel 4.3 berikut.

Tabel 4. 3 Hasil Proses Tokenisasi

Sebelum	**Sesudah**
penerapan word ngram sentiment analysis review metode support vector machine studi aplikasi sambaraabstractsambara application is an innovation from bapenda west java for motor vehicle tax services the sambara application expected can be provide efficiency effectiveness and service improvement the success of the application can be determined by conducting a sentiment review analysis sentiment analysis aims to detect polarity in the text in the form of negative or positive opinions using text mining at the text processing stage the word ngram feature is added as a word identification approach and	[penerapan, word, ngram, sentiment, analysis, review, metode, support, vector, machine, studi, aplikasi, sambaraabstractsambara, application, is, an, innovation, from, bapenda, west, java, for, motor, vehicle, tax, services, the, sambara, application, expected, can, be, provide, efficiency, effectiveness, and, service, improvement, the, success, of, the, application, can, be, determined, by, conducting, a, sentiment, review, analysis, sentiment, analysis, aims, to, detect, polarity, in, the, text, in, the, form, of, negative, or, positive, opinions, using, text, mining, at, the, text,

for classification it uses the support vector machine svm method this…	processing, stage, the, word, ngram, feature, is, added, as, a, word, identification, approach, and, for, classification, it, uses, the, support, vector, machine, svm, method, this, ...]

Terlihat dalam tabel 4.3 proses tokenisasi akan memecah kalimat menjadi sebuah kata-kata dalam bentuk list untuk memudahkan proses pre-processing selanjutnya. Kemudian proses *stemming* untuk mengubah kata yang memiliki imbuhan seperti awalan, akhiran, dan konfiks berubah menjadi kata dasar, penggunaan *library* dan modul nltk digunakan untuk proses *stemming* tersebut. Proses *stemming* seperti yang terlihat pada tabel 4.4.

Tabel 4. 4 Hasil Proses *Stemming*

Sebelum	**Sesudah**
[penerapan, word, ngram, sentiment, analysis, review, metode, support, vector, machine, studi, aplikasi, sambara abstractsambara, application, is, an, innovation, from, bapenda, west, java, for, motor, vehicle, tax, services, the, sambara, application, expected, can, be, provide, efficiency, effectiveness, and, service, improvement, the, success, of, the, application, can, be, determined, by, conducting, a, sentiment, review, analysis, sentiment, analysis, aims, to, detect, polarity, in, the, text, in, the, form, of, negative, or, positive, opinions, using, text, mining, at, the, text,	['terap', 'word', 'ngram', 'sentiment', 'analysis', 'review', 'metode', 'support', 'vector', 'machine', 'studi', 'aplikasi', 'sambara'],"['abstractsambara', 'application', 'innovation', 'bapenda', 'west', 'java', 'motor', 'vehicle', 'service', 'sambara', 'application', 'expect', 'provide', 'efficiency', 'effectiveness', 'service', 'improvement', 'success', 'application', 'determine', 'conduct', 'sentiment', 'review', 'analysis', 'sentiment', 'analysis', 'aim', 'detect', 'polarity', 'text', 'form', 'negative', 'positive', 'opinions', 'text', 'mine', 'text', 'process', 'stage', 'word', 'ngram', 'feature', 'add', 'word', 'identification', 'approach', 'classification', 'use',

processing, stage, the, word, ngram, feature, is, added, as, a, word, identification, approach, and, for, classification, it, uses, the, support, vector, machine, svm, method, this, ...]	'support', 'vector', 'machine', 'method',…]

Dapat dilihat pada tabel 4.4 bahwa proses *stemming* yang melibatkan modul sastrawi yang merupakan *library* berisi fungsi-fungsi yang dikhususkan untuk pemrosesan kata bahasa Indonesia. Adapula *stemmer factory* yang merupakan fungsi pemrosesan *stemming* pada setiap kata yang ada dikolom *title_abstract*, fungsi lain yang dipanggil yaitu fungsi yang bertugas dalam melakukan tokenisasi yaitu *preprocess(text)*. Fungsi tersebut akan membagi teks menjadi token dengan menggunakan fungsi *gensim.utils.simple_preprocess*. Kemudian, fungsi memeriksa setiap token untuk memastikan bahwa itu bukan *stopword*, dan panjang kata tidak lebih dari 3 kata. Fungsi preproses, yang didalamnya mengandung proses *stemming*, diterapkan ke kolom "*title_abstract*", barulah disimpan hasilnya kembali ke dalam DataFrame yaitu *df_pre*.

Memasuki tahapan *preprocessing* terakhir yakni proses normalisasi dataset, dengan menggunakan fungsi ‘to_list(strg_list), dengan menghapus item kosong dari daftar token dan membuat daftar baru token dengan semua token dari list_item yang memiliki panjang lebih dari 0, seperti pada tabel 4.5.

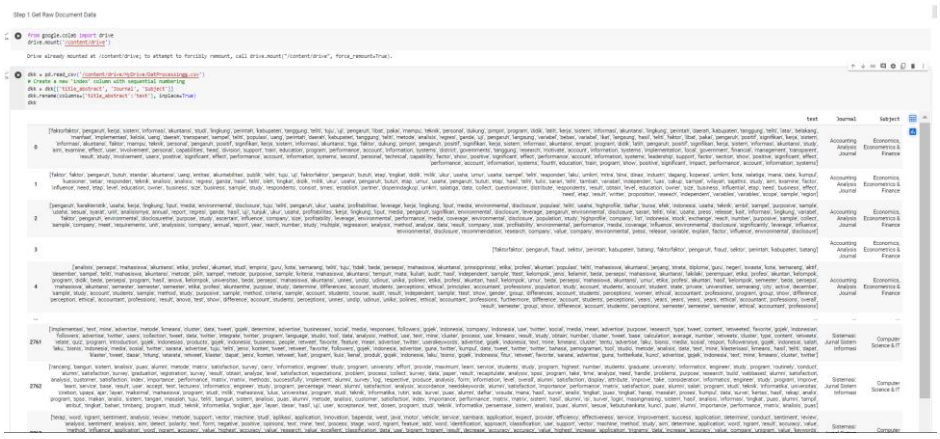
Tabel 4. 5 Hasil Proses Normalisasi

Sebelum	**Sesudah**
['terap', 'word', 'ngram', 'sentiment', 'analysis', 'review', 'metode', 'support', 'vector', 'machine', 'studi', 'aplikasi', 'sambara'],"['abstractsambara', 'application', 'innovation', 'bapenda', 'west', 'java', 'motor', 'vehicle', 'service', 'sambara', 'application', 'expect', 'provide', 'efficiency',	['terap', 'word', 'ngram', 'sentiment', 'analysis', 'review', 'metode', 'support', 'vector', 'machine', 'studi', 'aplikasi', 'sambara'],"['abstractsambara', 'application', 'innovation', 'bapenda', 'west', 'java', 'motor', 'vehicle', 'service', 'sambara', 'application', 'expect', 'provide', 'efficiency',

'effectiveness', 'service', 'improvement', 'success', 'application', 'determine', 'conduct', 'sentiment', 'review', 'analysis', 'sentiment', 'analysis', 'aim', 'detect', 'polarity', 'text', 'form', 'negative', 'positive', 'opinions', 'text', 'mine', 'text', 'process', 'stage', 'word', 'ngram', 'feature', 'add', 'word', 'identification', 'approach', 'classification', 'use', 'support', 'vector', 'machine', 'method',…]	'effectiveness', 'service', 'improvement', 'success', 'application', 'determine', 'conduct', 'sentiment', 'review', 'analysis', 'sentiment', 'analysis', 'aim', 'detect', 'polarity', 'text', 'form', 'negative', 'positive', 'opinions', 'text', 'mine', 'text', 'process', 'stage', 'word', 'ngram', 'feature', 'add', 'word', 'identification', 'approach', 'classification', 'use', 'support', 'vector', 'machine', 'method',…]

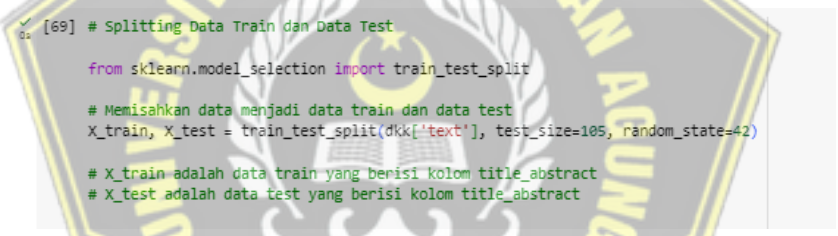
Terlihat pada tabel 4.5 bahwa fungsi *posttokenization_cleaning (unkn_input)* didefinisikan dan dipanggil guna menggabungkan langkah-langkah sebelumnya yaitu untuk membersihkan teks setelah proses tokenisasi. Fungsi tersebut memastikan *input unkn_input* adalah *string* atau daftar, dan untuk mengonversi *input* menjadi daftar, fungsi *to_list* digunakan. Sedangkan dalam menghapus item kosong dari daftar token, dengan fungsi *remove_empty_item*, yang seluruh fungsi tersebut di *apply* pada *df_pre* yang sebelumnya juga sudah dilakukan tahapan *stmming* dan *tokenisasi*. Proses normalisasi teks ini memiliki tujuan untuk melihat kembali hasil token untuk dilakukan pengaturan ulang agar data lebih rapi, dengan hasil akhirnya adalah daftar list yang menjadi topik-topik untuk dimodelkan nantinya.

Beralih ke tahapan selanjutnya yaitu menggabungkan kembali kolom *title_abstract* yang telah di *preprocessing* dalam bentuk *dataframe* bernama *df_pre* dengan kolom *Journal* dan *Subject* yang ada pada *dataframe* “dkk” menggunakan fungsi *concat*. Dalam menggabungkan 3 kolom tersebut terdapat satu kolom yang merupakan *primary key* nya yaitu *title_abstract*, perbedaannya adalah kolom *title_abstract* diganti dengan nama *text* yang isinya sudah berupa hasil dari proses *preprocessing* untuk memudahkan saat mencocokkan hasil pemodelan topik nantinya. Hasilnya adalah seperti yang terlihat pada gambar 4.2.



Gambar 4. 2 Menggabungkan Data Preprocessing dengan Kolom Journal dan Subject

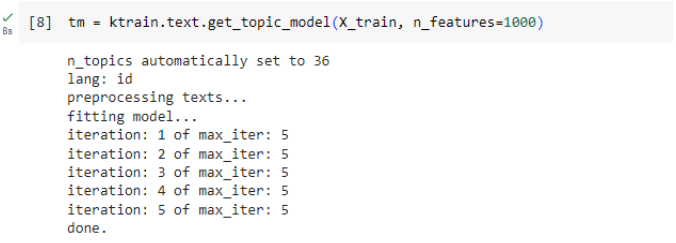
Dapat dilihat bahwa gambar 4.2 tersebut sudah menjadi sebuah *dataset* baru yang berisi 3 kolom utama yang siap digunakan untuk tahap selanjutnya. Tahapan berikutnya adalah melakukan *splitting* data menjadi X_*train* dan X_*test* , yang dapat dilihat pada gambar 4.3.



```
[69] # Splitting Data Train dan Data Test

     from sklearn.model_selection import train_test_split

     # Memisahkan data menjadi data train dan data test
     X_train, X_test = train_test_split(dkk['text'], test_size=105, random_state=42)

     # X_train adalah data train yang berisi kolom title_abstract
     # X_test adalah data test yang berisi kolom title_abstract
```

Gambar 4. 3 Proses *Splitting* Data

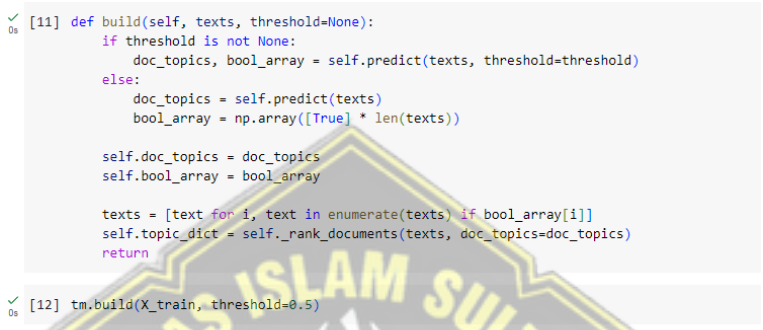
Terlihat bahwa gambar 4.3 tersebut menampilkan *source code* untuk membagi kolom *text* menjadi data *train* dan data *test*, pembagian jumlahnya meliputi data *train* sebanyak 2.661 dan data *test* sebanyak 105. Setelah didapatkan pembagian data *train* dan data *test*, maka selanjutnya adalah melakukan proses *Latent Dirichlet Allocation* (LDA) untuk mengetahui persebaran topik-topik yang ada pada data *train* maupun data *test* nya. Dengan hasilnya seperti terlihat pada gambar 4.4.



```
[8]  tm = ktrain.text.get_topic_model(X_train, n_features=1000)

     n_topics automatically set to 36
     lang: id
     preprocessing texts...
     fitting model...
     iteration: 1 of max_iter: 5
     iteration: 2 of max_iter: 5
     iteration: 3 of max_iter: 5
     iteration: 4 of max_iter: 5
     iteration: 5 of max_iter: 5
     done.
```

Gambar 4. 4 Hasil Proses LDA

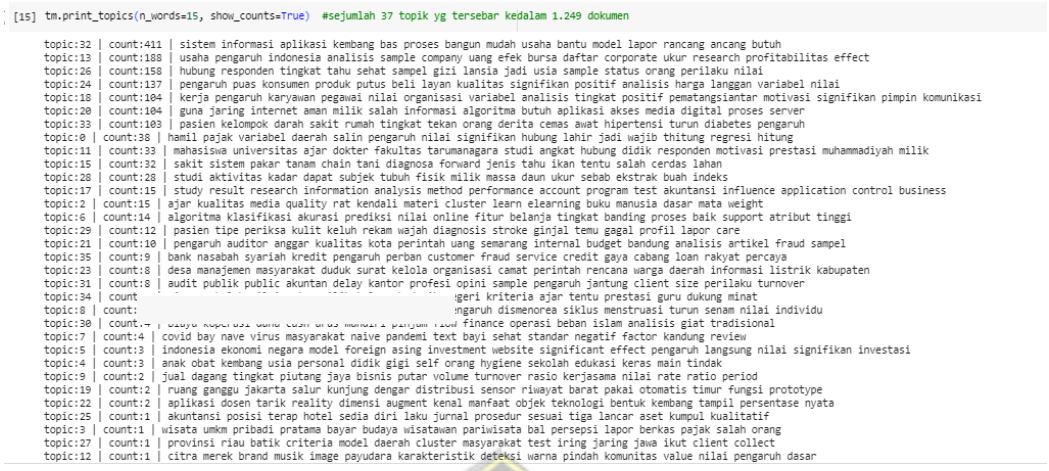
Dapat diamati dalam gambar 4.4 yakni pemanggilan kelas *topic_model* dengan hanya memakai data *text x_train* dan *n_features*=1000, sehingga fungsi lain yang dijalankan dibiarkan otomatis. Seperti pada *n_topics* pada X_*train* berhasil terbentuk sebanyak 36 *topic*, dengan iterasi maksimal sejumlah 5 kali. Dari 36 topik tersebut, tahapan selanjutnya yaitu mengidentifikasi penggalan kata pada topik-topiknya dengan menggunakan fungsi *print_topics* yang terlihat pada gambar 4.12.



```
[11] def build(self, texts, threshold=None):
         if threshold is not None:
             doc_topics, bool_array = self.predict(texts, threshold=threshold)
         else:
             doc_topics = self.predict(texts)
             bool_array = np.array([True] * len(texts))

         self.doc_topics = doc_topics
         self.bool_array = bool_array

         texts = [text for i, text in enumerate(texts) if bool_array[i]]
         self.topic_dict = self._rank_documents(texts, doc_topics=doc_topics)
         return

[12] tm.build(X_train, threshold=0.5)

     done.
```

Gambar 4. 5 Proses Build X_train

Terlihat pada gambar 4.5 bahwa pemanggilan fungsi *tm.build* pada 36 topik tadi diambil yang hanya memiliki ambang batas probabilitas topik sebesar 0.5 dan diatasnya, jadi untuk probabilitas topik dengan ambang batas dibawah 0.5 tidak termasuk. Lalu dari pengambilan *threshold* sebesar 0.5, ditampilkanlah topik dengan nomor berapa saja serta jumlah topik yang tersebar pada setiap dokumen dengan memanggil fungsi *print_topics* lagi, berbeda dari sebelumnya pada pemanggilan kali ini terdapat parameter *threshold* sehingga hasilnya sperti yang terlihat pada gambar 4.6.



```
[15] tm.print_topics(n_words=15, show_counts=True)  #sejumlah 37 topik yg tersebar kedalam 1.249 dokumen

topic:32 | count:411 | sistem informasi aplikasi kembang bas proses bangun mudah usaha bantu model lapor rancang ancang butuh
topic:13 | count:188 | usaha pengaruh indonesia analisis sample company uang efek bursa daftar corporate ukur research profitabilitas effect
topic:26 | count:158 | hubung responden tingkat tahu sehat sampel gizi lansia jadi usia sample status orang perilaku nilai
topic:24 | count:137 | pengaruh puas konsumen produk putus beli layan kualitas signifikan positif analisis harga langgan variabel nilai
topic:18 | count:104 | kerja pengaruh karyawan pegawai nilai organisasi variabel analisis tingkat positif pematangsiantar motivasi signifikan pimpin komunikasi
topic:20 | count:104 | guna jaring internet aman milik salah informasi algoritma butuh aplikasi akses media digital proses server
topic:33 | count:103 | pasien kelompok darah sakit rumah tingkat tekan orang derita cemas awat hipertensi turun diabetes pengaruh
topic:0 | count:38 | hamil pajak variabel daerah salin pengaruh nilai signifikan hubung lahir jadi wajib thitung regresi hitung
topic:11 | count:33 | mahasiswa universitas ajar dokter fakultas tarumanagara studi angkat hubung didik responden motivasi prestasi muhammadiyah milik
topic:15 | count:32 | sakit sistem pakar tanam chain tani diagnosa forward jenis tahu ikan tentu salah cerdas lahan
topic:28 | count:28 | studi aktivitas kadar dapat subjek tubuh fisik milik massa daun ukur sebab ekstrak buah indeks
topic:17 | count:15 | study result research information analysis method performance account program test akuntansi influence application control business
topic:2 | count:15 | ajar kualitas media quality rat kendali materi cluster learn elearning buku manusia dasar mata weight
topic:6 | count:14 | algoritma klasifikasi akurasi prediksi nilai online fitur belanja tingkat banding proses baik support atribut tinggi
topic:29 | count:12 | pasien tipe periksa kulit keluh rekam wajah diagnosis stroke ginjal temu gagal profil lapor care
topic:21 | count:10 | pengaruh auditor anggar kualitas kota perintah uang semarang internal budget bandung analisis artikel fraud sampel
topic:35 | count:9 | bank nasabah syariah kredit pengaruh perban customer fraud service credit gaya cabang loan rakyat percaya
topic:23 | count:8 | desa manajemen masyarakat duduk surat kelola organisasi camat perintah rencana warga daerah informasi listrik kabupaten
topic:31 | count:8 | audit publik public akuntan delay kantor profesi opini sample pengaruh jantung client size perilaku turnover
topic:34 | count                                                    geri kriteria ajar tentu prestasi guru dukung minat
topic:8 | count:                                                  ngaruh dismenorea siklus menstruasi turun senam nilai individu
topic:30 | count:4 | budaya kopi [illegible] finance operasi beban islam analisis giat tradisional
topic:7 | count:4 | covid bay nave virus masyarakat naive pandemi text bayi sehat standar negatif factor kandung review
topic:5 | count:3 | indonesia ekonomi negara model foreign asing investment website significant effect pengaruh langsung nilai signifikan investasi
topic:4 | count:3 | anak obat kembang usia personal didik gigi self orang hygiene sekolah edukasi keras main tindak
topic:9 | count:2 | jual dagang tingkat piutang jaya bisnis putar volume turnover rasio kerjasama nilai rate ratio period
topic:19 | count:2 | ruang ganggu jakarta salur kunjung dengar distribusi sensor riwayat barat pakai otomatis timur fungsi prototype
topic:22 | count:2 | aplikasi dosen tarik reality dimensi augment kenal manfaat objek teknologi bentuk kembang tampil persentase nyata
topic:25 | count:1 | akuntansi posisi terap hotel sedia diri laku jurnal prosedur sesuai tiga lancar aset kumpul kualitatif
topic:3 | count:1 | wisata umkm pribadi pratama bayar budaya wisatawan pariwisata bal persepsi lapor berkas pajak salah orang
topic:27 | count:1 | provinsi riau batik criteria model daerah cluster masyarakat test iring jaring jawa ikut client collect
topic:12 | count:1 | citra merek brand musik image payudara karakteristik deteksi warna pindah komunitas value nilai pengaruh dasar
```

Gambar 4. 6 Hasil Persebaran Topik Berdasarkan *Threshold*

Terlihat dari gambar 4.6 bahwa dengan *threshold* 0.5, topik yang teridentifikasi dengan jumlah 36 berkurang menjadi 31 topik dengan jumlah persebaran sejumlah 1.495 dokumen. Selanjutnya, dari hasil tersebut dapat dilakukan *doc topic mapping* dengan mendeklarasikan fungsi *get_docs* dengan langkah awal memeriksa proses *build* lalu memetakan dokumen berdasarkan topik-topiknya, fungsi tersebut dapat diberikan parameter dengan meng*input*kan *topic_ids, doc_ids, rank*, jika hanya ingin melihat dokumen tertentu bisa di*input*kan. Hasil pemanggilan fungsi tersebut berbentuk list (daftar), sehingga perlu diubah menjadi *dataset* yang dapat diamati pada gambar 4.7, agar mudah digunakan dalam proses tahapan selanjutnya.

```
[17] all_docs=tm.get_docs(rank=True)
     all_docs = pd.DataFrame(all_docs)

     all_docs.to_csv('doctopik.csv', index=False)

[19] doctopic=pd.read_csv('doctopik.csv')

[20] doctopic
```

Gambar 4. 7 Hasil *Doc Topic* Mapping dalam bentuk Dataset Baru

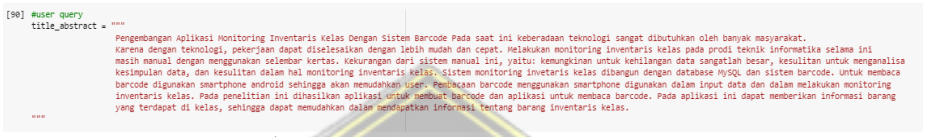
Dapat dilihat pada gambar 4.7 hasil *dataset* yang memuat pemetaan dokumen berdasarkan topik-topiknya yang dikelompokkan pada masing-masing kolom. Setelah itu, melakukan fungsi *pd.merge dataset doc topic mapping* dengan *dataset* "dpp" yang mana didalam *dataset* "dpp" memuat kolom *text*, *journal*, dan *subject* dengan metode *inner* pada kolom *text* sebagai *primary key* nya yang terlihat pada gambar 4.18.

Gambar 4. 8 *Doc Topic Dataset*

Dapat diamati pada gambar 4.8 sebuah *dataset* yang dipersiapkan untuk tahapan lanjutan yakni pembuatan *recommender machine*. Dalam pembuatan *recommender machine* menggunakan metode K-*Nearest Neighborss* (KNN) dengan terlebih dahulu mempersiapkan fungsi *get_doctopics* yaitu fungsi yang berguna dalam mengembalikan distribusi probabilitas topik pada semua dokumen yang ada didalam model. Distribusi probabilitas topik ini menunjukkan seberapa besar probabilitas bahwa dokumen tersebut terkait yang merujuk pada probabilitas setiap topik untuk dalam model yang hasilnya dalam bentuk *np. array*. Fungsi *get_doctopics* dapat diisi parameter yang hanya menuliskan *topic_ids* dan *doc_ids* tertentu saja, namun jika parameter tidak diisi, maka hasilnya menampilkan semua dokumen yang terkandung didalam *topic model* dengan setiap baris mewakili dokumen dan setiap kolom mewakili topik yang berbeda-beda.

Kemudian tahapan selanjutnya adalah meng-implementasikan metode K-*Nearest Neighbors* (KNN) kedalam fungsi *train recommender*. Dalam mengimplementasikan KNN pada metode *train_recommender* yang melibatkan modul *sklearn.neighborss* untuk meng*import* kelan *nearest neighborss*, didalam fungsi diatas ditentukan parameter n_*neighbors*nya yaitu banyaknya tetangga yang akan dilatih sebanyak 20. Selain itu, penggunaan *metric* sebagai penghitungan jarak antar tetangga menggunakan *metric minkowski* dengan p=2 (bernilai positif). Baris selanjutnya adalah objek *rec* dari kelas *NearestNeighborss* dengan parameter yang dapat diubah atau secara *default*, beralih ke fungsi *get_doctopics* yang dipanggil untuk menjadi fitur dalam train model KNN. Kemudian objek *rec* yang merupakan

model KNN dilatih dengan objek *probs* dari distribusi probabilitas topik yang sebelumnya sudah dijalankan. Hasil latih disimpan kedalam atribut bernama *recommend*. Pada sel yang terpisah metode *train recommend* dijalankan dan berhasil. Tahapan berikutnya adalah melakukan *journal recommendation* dengan tahapannya adalah meng*input*kan terlebih dahulu *user query* berupa *title* dan *abstract* yang diambil dari dataset kita, dalam hal ini menggunakan data *training* untuk mengujinya, yang dapat dilihat pada gambar 4.9.



Gambar 4. 9 *Input Query title abstract*

Terlihat didalam gambar 4.9 bahwa *query* yang di*input*kan dalam bentuk *title abstract* berbentuk teks biasa. Setelah itu, barulah menuliskan metode *recommend* yang sebelumnya sudah dijalankan berisi data latih KNN.

Jumlah tetangga akan diatur menjadi n jika jumlah rekomendasi (n) lebih besar dari jumlah tetangga yang ditentukan (*n_neighbors*). Dalam variabel x_test, topik distribusi dokumen (*doc_topic*) atau teks *input (text*) harus dimasukkan, teks tersebut akan digunakan sebagai dasar untuk membuat distribusi topik dokumen menggunakan metode *predict*. Metode *kneighborss* objek rekomendasi (model KNN) digunakan untuk menemukan indeks tetangga terdekat dari input *x_test.* Fungsi *return_distance=False* menunjukkan bahwa jarak antara tetangga dan *input* tidak diperlukan. Selanjutnya, dokumen yang sesuai dari kumpulan dokumen dipilih dengan menggunakan indeks dokumen yang ditemukan sebagai tetangga terdekat, dokumen yang dipilih untuk digunakan sebagai rekomendasi akan dikembalikan. Langkah berikutnya yaitu proses *identify topic* dengan melakukan pemanggilan metode *recommend* dan hasilnya dapat dilihat pada gambar 4.10.

Gambar 4. 10 Pemanggilan Metode recommend dan Hasilnya

Terlihat pada gambar 4.10 yang menunjukkan pemanggilan serta hasil dari metode *recommend*, dimana hasil tersebut menampilkan 10 rekomendasi mulai dari *text, doc_id, topic_proba,* dan *topic_id* dari masing-masing dokumen yang direkomendasikan dari *input query* yang sebelumnya telah di*input*. Tahapan selanjutnya yaitu *doc topic filtering* dimana hasil dari rekomendasi tadi dipetakan kedalam topik-topiknya serta dilakukan *filtering* dan diurutkan berdasarkan *topic proba*, dan *topic_id*, terlihat pada gambar 4.11.

```
[93] #doc topic filtering
     docs_filtered = [doc for doc in recommendations if doc['topic_proba']]

     # Urutkan dokumen berdasarkan nilai probabilitas topik secara ascending
     sorted_docs = sorted(docs_filtered, key=lambda x: x['topic_id'], reverse=False)

     # Buat DataFrame baru dari dokumen yang telah diurutkan
     recom = pd.DataFrame(sorted_docs)

     # Tampilkan DataFrame baru
     print(recom)
```

Gambar 4. 11 Penulisan Fungsi *Doc Topic Filtering*

Dapat diamati pada gambar 4.11 dengan menuliskan fungsi untuk mem-*filter* daftar rekomendasi yang dihasilkan dari metode *recommend*. Kemudian

hasil dari *doc topic filtering* tersebut dilakukan *topic article* yaitu mencocokkan 10 rekomendasi yang dikelompokkan kedalam kolom *text*, *topic_id*, *doc_id*, dan *topic_proba* termasuk kedalam jenis jurnal yang mana. Penulisan fungsi *topic article* dapat dilihat pada gambar 4.12.

```
#Topic Article
def match_doc_id(doc_id):
    if doc_id in dokk["text"].tolist():
        return dokk.loc[dokk["text"] == doc_id, "Journal"].iloc[0]
    else:
        return None

recom["Jurnal"] = recom["text"].apply(match_doc_id)
print(recom)
```

Gambar 4. 12 Penulisan Fungsi *Topic Article*

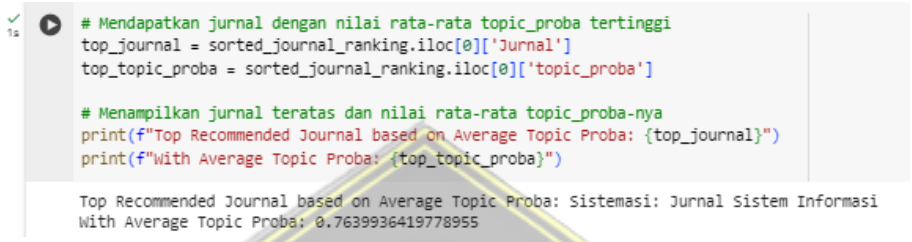
Dapat dilihat di gambar 4.12 mengenai penulisan fungsi *topic article* yang menggunakan metode *match_doc_id* untuk memeriksa *doc_id*, lalu dari *doc_id* tersebut dicocokkan dengan *loc text* yang sama dari *dataset doc topic mapping* yang berasal dari data *train* tadi untuk mencari jurnal yang cocok dengan *doc_id* tersebut dengan fungsi *iloc*. Setelah itu, mendeklarasikan fungsi *apply* terhadap *match_doc_id* pada *dataframe* yang baru berisikan 10 rekomendasi yang lengkap dengan jurnalnya. Langkah berikutnya adalah *journal ranking*, dari hasil *dataframe recom* dengan mencari rata-rata dari *topic probabilistic* setiap jurnal yang direkomendasikan. Adapun penulisan *source code journal ranking* terlihat pada gambar 4.13.

```
[95] #Journal Ranking
     filtered_recommendations_list = []
     for topic_id_to_filter in recom['topic_proba'].unique():
         filtered_recommendations = recom[
             (recom['topic_proba'] == topic_id_to_filter)
         ]
         if not filtered_recommendations.empty:
             filtered_recommendations_list.append(filtered_recommendations)

     # Concatenate the filtered recommendations for all topic_ids
     filtered_recommendations_all = pd.concat(filtered_recommendations_list)

     # Calculate the average topic_proba for each Journal
     journal_ranking = filtered_recommendations_all.groupby('Jurnal')['topic_proba'].mean().reset_index()

     # Sort the journal ranking by average topic_proba in descending order
     sorted_journal_ranking = journal_ranking.sort_values(by='topic_proba', ascending=False)

     # Display the resulting DataFrame
     print("Journal Ranking :")
     print(sorted_journal_ranking)
```

Gambar 4. 13 Penulisan Fungsi *Journal Ranking*

Dapat terlihat pada gambar 4.13 bahwa *journal ranking* dihitung melalui rata-rata *topic_probabilistic* setiap jurnal, jika ada jurnal yang sama maka fungsi

unique akan mengambil *topic_proba* yang tertinggi dari jurnal yang sama tersebut untuk dihitung *mean()*, kemudian barulah diambil setiap kategori jurnal berbeda dengan *topic_proba* yang berbeda pula. Setelah itu dilakukan *sort* untuk melihat *topic_proba* tertinggi dari setiap jurnal. Tahapan terakhir adalah menampilkan rekomendasi jurnal teratas yang *topic_probalistic* tertinggi, seperti yang dapat diamati pada gambar 4.14.



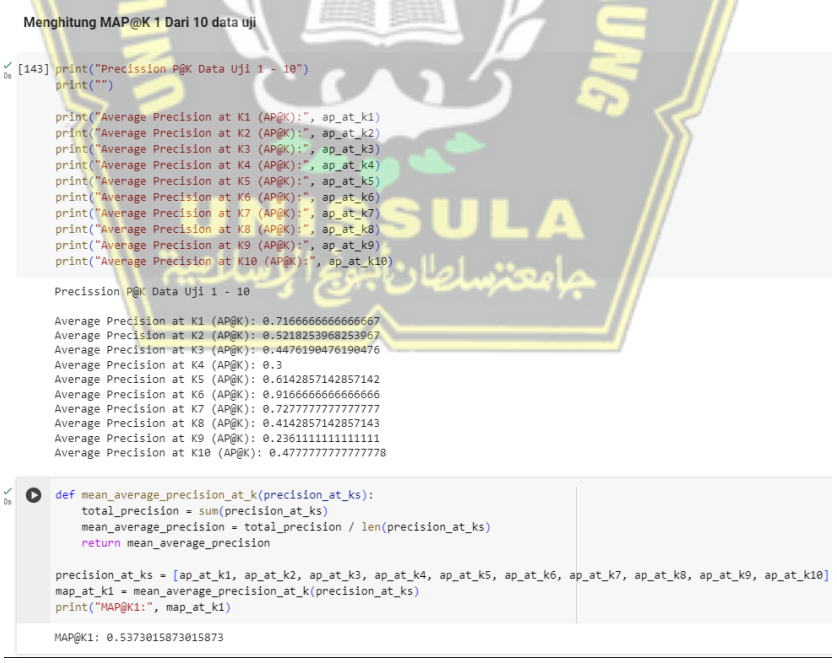
Gambar 4. 14 Hasil *Top Recommended Journal*

Terlihat pada gambar 4.14 yang menampilkan hasil jurnal yang paling direkomendasikan berdasarkan *input query* nya, adalah jurnal yang memiliki *topic probabilistic* sebesar 0,7639. Berdasarkan hasil uji X_*train* yang telah dilakukan proses dari awal sampai akhir menghasilkan jurnal dengan *topic probabilistic* sebesar 0,7639 yang mana hasil ini sudah relevan dengan *title abstract* yang di*input*kan. Karena dilihat dari topik-topik yang muncul dalam *title abstract* hampir sama dengan topik yang muncul pada artikel di salah satu jurnal Sistemasi. Untuk membuktikan apakah sistem rekomendasi jurnal tersebut efektif dalam menghasilkan rekomendasi jurnal yang relevan, maka dilakukan *evaluation metrics* dengan menghitung *Precision @ K* (P@K) pada setiap data uji X_*test* dengan jumlah 100 data. Dengan setiap data uji diambil 10 rekomendasi teratasnya, lalu dihitung P@K rumus nomor (2) dan *Average Precision* @K (AP@K) dengan rumus nomor (3), seperti yang dapat dilihat pada gambar 4.15.

```
def precision_at_k(ground_truth, recommended_items, k):
    # Menghitung jumlah item relevan di antara K item yang direkomendasikan
    relevant_recommended = [item for item in recommended_items[:k] if item in ground_truth]
    # Menghitung precision@k
    p_at_k = len(relevant_recommended) / k
    return p_at_k

ground_truth = ['Window of Nursing Journal', 'Care : Jurnal Ilmiah Ilmu Kesehatan', 'Health and Medical Journal']  # jurnal yang relevan
recommended_items = ['Window of Nursing Journal', 'Window of Nursing Journal', 'Care : Jurnal Ilmiah Ilmu Kesehatan',
                     'Tarumanagara Medical Journal', 'Tarumanagara Medical Journal', 'Window of Nursing Journal',
                     'Health and Medical Journal', 'Health and Medical Journal', 'Health and Medical Journal',
                     'Window of Nursing Journal']  # Daftar 10 rekomendasi teratas

print("Precission P@K Data Uji 1")
print("")

for k in range(1, 11):
    p_at_k = precision_at_k(ground_truth, recommended_items, k)
    print(f"Precission @K={k}:", p_at_k)

def average_precision_at_k(relevant_items, recommended_items, k):
    rel = [1 if item in ground_truth else 0 for item in recommended_items[:k]]
    total_rel = sum(rel)
    ap_at_k1 = sum(rel[i] / (i + 1) for i in range(k)) / min(k, len(relevant_items))
    return ap_at_k1

k = 10  # K adalah 10 untuk menghitung AP@K

ap_at_k1 = average_precision_at_k(ground_truth, recommended_items, k)
print("")
print("Average Precision at K1 (AP@K):", ap_at_k1)
```

Gambar 4. 15 Menghitung P@K Pada Data Uji 1

Dapat diamati pada gambar 4.15 yang menampilkan fungsi berupa rumus penghitungan AP@K pada 10 rekomendasi data uji 1 dengan *ground truth* nya adalah jurnal dengan menghasilkan 0,7166. Seterusnya juga dilakukan untuk data uji 2-10, untuk mendapatkan *Mean Average Precision* @K1 dengan penulisan fungsi pada gambar 4.16 berikut.



Menghitung MAP@K 1 Dari 10 data uji

```
[143] print("Precission P@K Data Uji 1 - 10")
      print("")

      print("Average Precision at K1 (AP@K):", ap_at_k1)
      print("Average Precision at K2 (AP@K):", ap_at_k2)
      print("Average Precision at K3 (AP@K):", ap_at_k3)
      print("Average Precision at K4 (AP@K):", ap_at_k4)
      print("Average Precision at K5 (AP@K):", ap_at_k5)
      print("Average Precision at K6 (AP@K):", ap_at_k6)
      print("Average Precision at K7 (AP@K):", ap_at_k7)
      print("Average Precision at K8 (AP@K):", ap_at_k8)
      print("Average Precision at K9 (AP@K):", ap_at_k9)
      print("Average Precision at K10 (AP@K):", ap_at_k10)

      Precission P@K Data Uji 1 - 10

      Average Precision at K1 (AP@K): 0.7166666666666667
      Average Precision at K2 (AP@K): 0.5218253968253967
      Average Precision at K3 (AP@K): 0.4476190476190476
      Average Precision at K4 (AP@K): 0.3
      Average Precision at K5 (AP@K): 0.6142857142857142
      Average Precision at K6 (AP@K): 0.9166666666666666
      Average Precision at K7 (AP@K): 0.7277777777777777
      Average Precision at K8 (AP@K): 0.4142857142857143
      Average Precision at K9 (AP@K): 0.2361111111111111
      Average Precision at K10 (AP@K): 0.4777777777777778
```

```
def mean_average_precision_at_k(precision_at_ks):
    total_precision = sum(precision_at_ks)
    mean_average_precision = total_precision / len(precision_at_ks)
    return mean_average_precision

precision_at_ks = [ap_at_k1, ap_at_k2, ap_at_k3, ap_at_k4, ap_at_k5, ap_at_k6, ap_at_k7, ap_at_k8, ap_at_k9, ap_at_k10]
map_at_k1 = mean_average_precision_at_k(precision_at_ks)
print("MAP@K1:", map_at_k1)

MAP@K1: 0.5373015873015873
```

Gambar 4. 16 Hasil MAP@K 1

Terlihat pada gambar 4.16 tersebut menampilkan MAP@K 1 yang dihitung dari rumus MAP@K nomor (4) yang di*input*kan kedalam *source code* diatas. Sehingga dihasilkan MAP@K 1 untuk jumlah 10 data uji pertama sebesar 0,8263.

Proses evaluasi ini dilakukan secara berurutan sampai pada data uji ke-100, dimana hasil akhir dari evaluasi MAP@K tersebut sejumlah 10 MAP@K yang didapatkan dari 100 data X_*test* dapat diamati pada gambar 4.16.

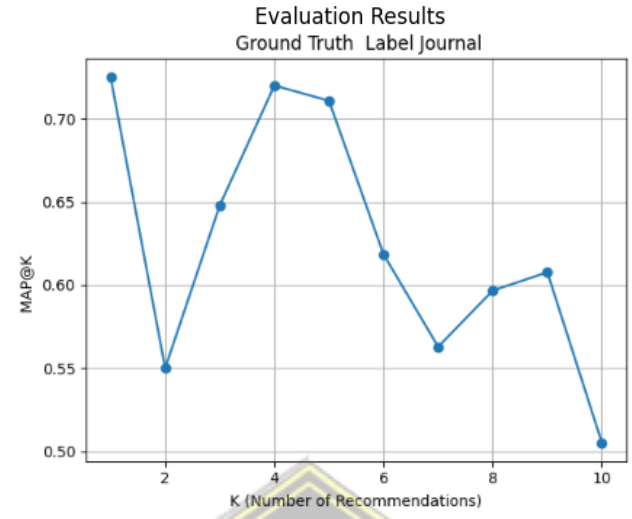
Terlihat pada gambar 4.19 bahwa MAP@K dari 1-10 memiliki rentang hasil akurasi di angka 0,5051 - 0,7250 atau dapat dikatakan rentang 50% - 72% yang mana hasil ini tergolong cukup baik. Disamping melakukan uji data *test* pada *ground truth journal* yang diujikan, dilakukan pula untuk melihat *ground truth subject* menggunakan model dan data *test* yang sama yaitu sejumlah 100, dengan hasil MAP@K 1-10 dapat dilihat pada gambar 4.19 berikut.

Seperti yang diamati pada gambar 4.31yang merupakan hasil akhir MAP@K 1-10 yang diambil dari *ground truth subject* yang hasilnya rata-rata di angka 0,9521 – 0,9763 atau dapat disebut rentang 95% - 97%, hasil tersebut tergolong baik sekali karena presentase nya mencapai di angka lebih dari 90%. Untuk melihat perbandingan hasil evaluasi uji model dengan MAP@K *ground truth journal* dengan MAP@K *ground truth subject* dapat diamati pada tabel 4.1 berikut.

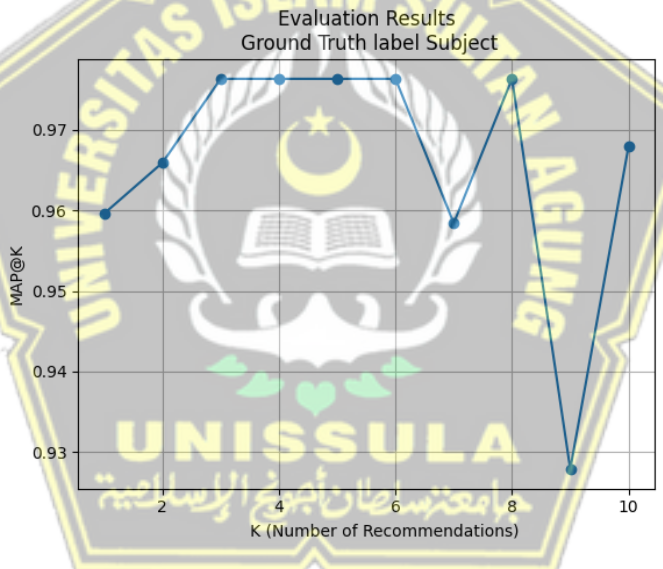
Tabel 4. 6 Hasil *Mean Average Precision* Rekomendasi

Input	**Ground Truth**	**MAP@K**									
		MAP@K1	**MAP@K2**	**MAP@K3**	**MAP@K4**	**MAP@K5**	**MAP@K6**	**MAP@K7**	**MAP@K8**	**MAP@K9**	**MAP@K10**
Title + Abstract	Journal	0,7250	0,5500	0,6478	0,7198	0,7106	0,6183	0,5629	0,5966	0,6076	0,5051
	Subject	0,9596	0,9659	0,9763	0,9763	0,9763	0,9763	0,9584	0,9763	0,9521	0,9679

Dapat dilihat melalui tabel 4.6 bahwa pada *ground truth journal* terdapat MAP@K paling tinggi ada di MAP@K 1 dengan hasilnya adalah 0,7250. Lain halnya dengan MAP@K dengan *ground truth subject* yang memiliki akurasi tinggi diangka 0,9763. Berikut grafik dari MAP@K dengan *ground truth journal* dan *ground truth Subject* yang dapat diamati pada gambar 4.21 dan 4.22.



Gambar 4. 17 Hasil Pengujian Rekomendasi Journal



Gambar 4. 18 Hasil Pengujian Rekomendasi *Subject*

Dapat dilihat pada gambar 4.18 dengan hasil evaluasi berdasarkan *ground truth* jurnal akurasinya tinggi saat pertama kali pengujian lalu seterusnya naik turun dalam rentang akurasi 50%-72% dan yang paling tinggi adalah 72% pada MAP@K ke1. Berbeda dengan *ground truth* dari label *Subject* yang terlihat pada gambar 4.22 memiliki akurasidi rentang 90% dan tertinggi adalah 97%, hal ini disebabkan karena label *subject* jangkauannya lebih luas dan dapat memuat berbagai macam jurnal yang berbeda pada setiap *subject*nya. Sehingga dalam merekomendasikan akurat dan relevan dibandingkan saat merekomendasikan jurnal.

4.2 Hasil Implementasi Antarmuka

Setelah dilakukan tahapan pemodelan selesai, tahap berikutnya yaitu tahap implementasi sistem pada *platform website*, dalam hal ini memakai *streamlit*. Antarmuka tersebut dikembangkan dengan merujuk pada desain yang telah dibuat sebelumnya yakni *wireframe* sistem yang tercantum dalam bab III.

4.2.1 Halaman Awal Sistem Rekomendasi

Dalam implementasi dari desain antarmuka halaman depan sistem rekomendasi jurnal, bagian *appbar* terdapat *title* dari aplikasi. Bagian bawah *title* terdapat *input text area* untuk meng*input*kan teks berupa judul dan abstrak suatu artikel dan *button* bertuliskan "Temukan Rekomendasi" guna memproses *input*nya. Halaman awal Sistem Rekomendasi dapat diamati pada gambar 4.34.

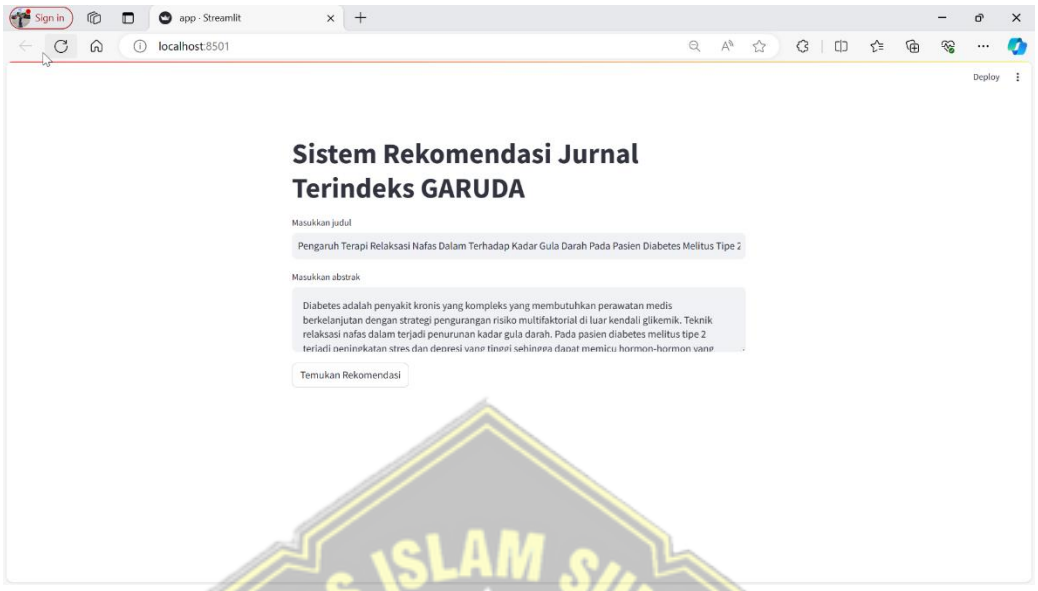
Gambar 4. 19 Halaman Awal Sistem

Terlihat di gambar 4.19 tersebut, adalah tampilan awal yang akan dilihat oleh pengguna yang mengakses. Pada tampilan tersebut, pengguna bisa langsung memberikan *input* teks kedalam *text area* yang ada.

4.2.2 Halaman Hasil Sistem Rekomendasi

Bentuk implementasi dari desain antarmuka halaman hasil sistem rekomendasi, hampir sama dengan halaman awal sistem. Perbedaannya adalah dalam halaman hasil terdapat hasil rekomendasi jurnal dari *input* yang telah diproses oleh sistem, untuk demo aplikasi dengan mengetikkan *input* pada kolom

text area seperti yang terlihat dalam gambar 4.20.

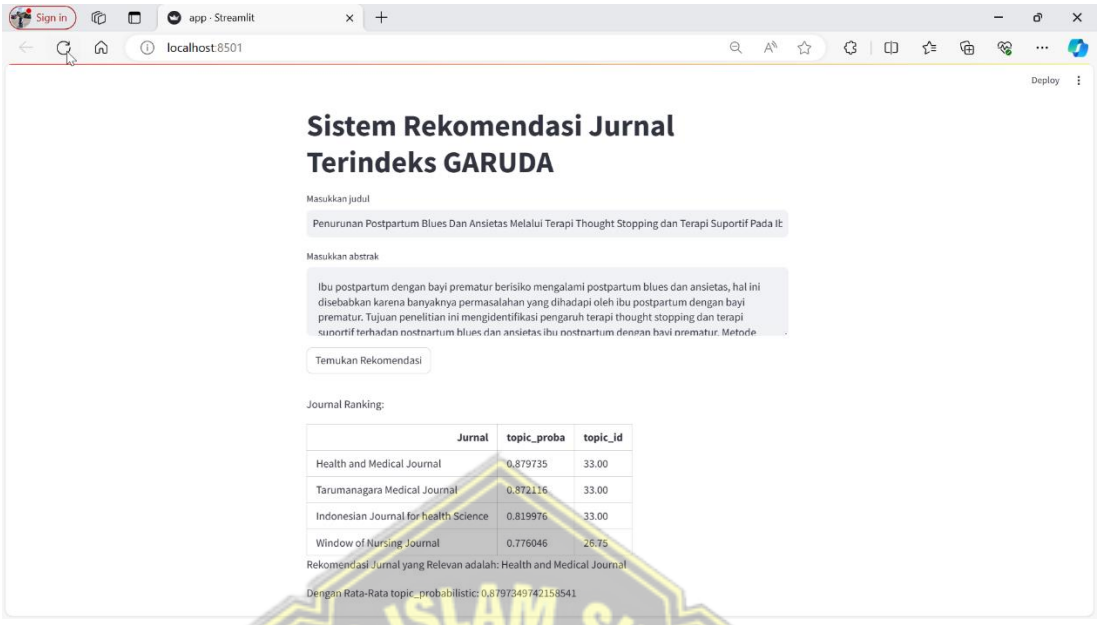


Gambar 4. 20 Meng*input*kan Judul dan Abstrak Artikel

Dapat diamati didalam gambar 4.20 tersebut adalah demo aplikasi dengan meng*input*kan salah satu artikel yang didapat dari *platform* GARUDA, dan hasil rekomendasinya dapat dilihat pada gambar 4.34.

Gambar 4. 21 Demo Sistem Rekomendasi ke-1

Dilihat pada gambar 4.21, hasil rekomendasi yang diberikan dinilai relevan dengan *input* yang diberikan, karena artikel tersebut membahas mengenai dampak psikologi. Selanjutnya mencoba untuk meng*input*kan judul dan abstrak artikel lainnya, dengan demo aplikasi yang ada pada gambar 4.22.



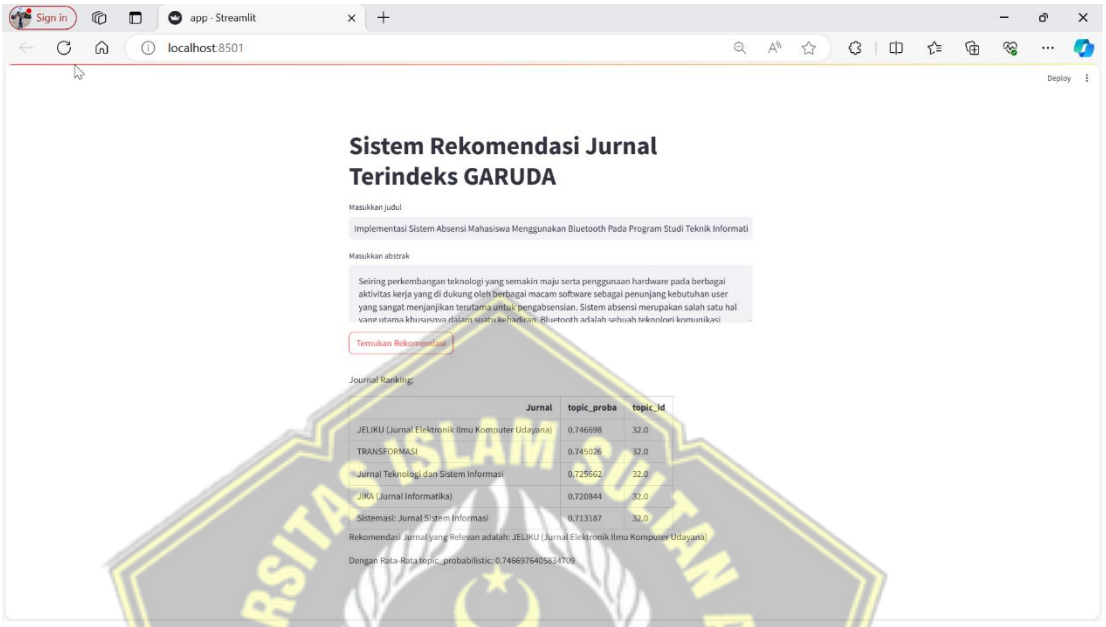
Gambar 4. 22 Demo Sistem Rekomendasi ke-2

Pada gambar 4.22 hasil rekomendasi dapat menampilkan hasil yang kurang relevan dengan *input* artikel seputar kesehatan, namun hasil rekomendasi jurnal yang seharusnya adalah jurnal Care. Kemudian mencoba lagi untuk meng*input*kan judul dan abstrak dari artikel lain yang dapat diamati dalam gambar 4.22.

Gambar 4. 23 Demo Sistem Rekomendasi ke-3

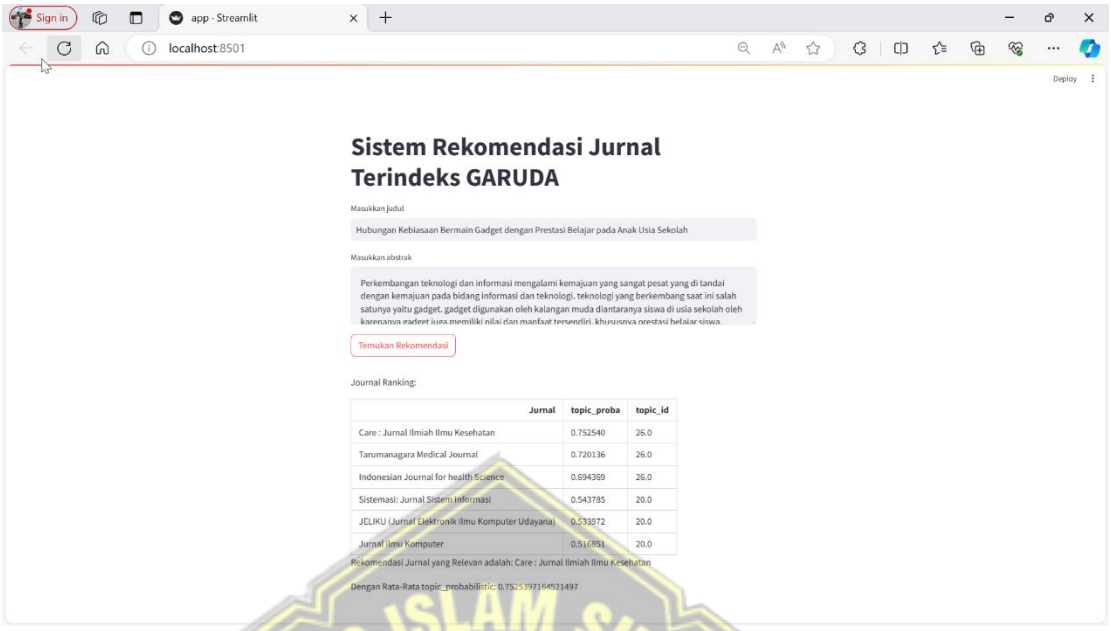
Terlihat di gambar 4.23, bahwa sistem rekomendasi memberikan hasil rekomendasi yang kurang relevan dengan *input* yang diberikan oleh pengguna, karena hasil rekomendasinya mengenai seputar ilmu informatika, padahal

seharusnya termasuk kedalam ilmu computer yang mrupakan rekomendasi ke-5. Berikutnya adalah percobaan kembali meng*input*kan judul dan abstrak artikel lainnya yang dapat dilihat pada gambar 4.24.



Gambar 4. 24 Demo Sistem Rekomendasi ke-4

Pada gambar 4.24, hasil rekomendasi jurnal dari sistem rekomendasi tersebut menghasilkan rekomendasi jurnal yang relevan karena *input*nya dan hasil rekomendasinya sudah sesuai. Selanjutnya percobaan lagi untuk melihat kinerja sistem rekomendasi tersebut, yang dapat dilihat pada gambar 4.25.



Gambar 4. 25 Demo Sistem Rekomendasi ke-5

Dapat diamati pada gambar 4.25, sistem rekomendasi memberikan rekomendasi jurnal yang dinlai relevan dengan *input* artikel yang membahas bidang Kesehatan. Sehingga, dari lima kali percobaan demo aplikasi, sistem rekomendasi tersebut dapat bekerja dalam memproses dan memberikan rekomendasi jurnal yang relevan dengan *input* yang dilakukan oleh pengguna.

4.3 Hasil Pengujian Sistem

Berdasarkan tahap pengujian sistem rekomendasi jurnal perlu dilakukan pengujian *black box*, pengujian ini bertujuan untuk mengetahuo kinerja dari sistem apakah dapat berfungsi dengan baik sesuai dengan yang diharapkan dan hasilnya dapat terlihat dalam *output*. Dengan rencana yang diimplementasikan adalah *input*, hasil yang diharapkan, *output* dan kesimpulan, berikut merupakan hasil pengujian *black box* yang dituliskan dalam tabel 4.2.

Tabel 4. 7 Tabel Pengujian Sistem Rekomendasi

Input	**Hasil yang diharapkan**	***Output***	**Kesimpulan**
Pengaruh Terapi Relaksasi Nafas dalam Terhadap	Menampilkan Rekomendasi Jurnal *Window of*	Menampilkan Rekomendasi Jurnal Care :	Hasil Kurang Relevan dan Kurang Sesuai

Kadar Gula Darah..	*Nursing Journal*	Jurnal Ilmiah Ilmu Kesehatan	
Penurunan Postpartum Blues dan Ansietas Melalui Terapi Thought Stopping...	Menampilkan Rekomendasi Jurnal *Health and Medical Journal*	Menampilkan Rekomendasi Jurnal Care : Jurnal Ilmiah Ilmu Kesehatan	Hasil Kurang Relevan dan Kurang Sesuai
Sistem Informasi Penilaian Kinerja Dosen dengan MVC Framework menggunakan Simple……	Menampilkan Rekomendasi Jurnal *Building of Informatics, Tchnology and Science* (BITS)	Menampilkan Rekomendasi Jurnal Ilmu Komputer	Hasil Kurang Relevan dan Kurang Sesuai
Hubungan Kebiasaan Bermain Gadget dengan Prestasi Belajar pada Anak Usia Sekolah……	Menampilkan Rekomendasi Jurnal Care : Jurnal Ilmiah Ilmu Kesehatan	Menampilkan Rekomendasi Jurnal Care : Jurnal Ilmiah Ilmu Kesehatan	Hasil Relevan dan Sesuai
Implementasi Sistem Absensi Mahasiswa Menggunakan Bluetooth pada Program Studi Teknik Informatika……..	Menampilkan Rekomendasi Jurnal JELIKU (Jurnal Elektronik Ilmu Komputer Udayana)dan Keuangan	Menampilkan Rekomendasi Jurnal JELIKU (Jurnal Elektronik Ilmu Komputer Udayana) dan Keuangan	Hasil Relevan dan Sesuai

Dapat diamati dalam tabel 4.3 bahwa hasil 5 kali pengujian *black box* yang sudah dilakukan, terdapat hanya dua hasil uji yang merekomendasikan jurnal yang relevan dengan judul dan abstrak suatu artikel, namun tiga hasil uji lainnya menghasilkan rekomendasi jurnal yang kurang relevan dengan *input* yang diberikan oleh *user*. Pengujian *black box* ini meng*input*kan judul dan abstrak yang tidak ada didalam data model yang sudah dilatih, sehingga dapat disimpulkan bahwa setiap judul dan abstrak artikel yang di *input*kan belum tentu sesuai dengan rekomendasi jenis jurnal yang ada.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil dari penelitian ini, dapat disimpulkan bahwa implementasi metode *Latent Dirichlet Allocation* (LDA) dan K-*Nearest Neighborss* (KNN) pada sistem rekomendasi jurnal terindeks GARUDA ini berjalan dengan baik saat memodelkan topik dan memetakannya kedalam beberapa jenis yaitu *topic_id, topic_proba* dan *doc_id*. Sedangkan dalam memberikan informasi mengenai kesesuaian topik probabilitas berdasarkan judul dan abstrak artikel yang di*input*kan dengan metode KNN yang terbilang sudah cukup baik dalam memberikan rekomendasi jurnal yang relevan. Hal ini berdasarkan data pengujian evaluasi matriks dengan metode *Mean Average Precision* @K dalam menguji 100 data *test* memiliki akurasi mencapai 0,7250 atau 72%. Sehingga hasil ini belum sesuai dengan yang diharapkan, faktor penyebabnya bisa jadi karena adanya data *test* yang diujikan masih ada beberapa yang tercampur dengan Bahasa inggris serta adanya persamaan kata pada judul dan abstrak dari satu jurnal yang sama, namun tidak sesuai dengan topik bahasan.

5.2 Saran

Berdasarkan penelitian yang telah dilakukan, Peneliti memiliki saran untuk pengembangan maupun penelitian yang akan dating, yaitu sebagai berikut:

1. Meningkatkan tingkat akurasi *precision* dengan mengambil *dataset title abstract* yang hanya memakai Bahasa Indonesia (tidak bercampur dengan Bahasa lainnya).
2. Mengklasifikasikan *topic_id* menjadi sebuah topik yang jelas, misalnya topik Kesehatan, topik Sistem Informasi, dan lain sebagainya, dan bukan hanya *id topic* nya saja.
3. Pada penelitian ini hanya fokus memberikan rekomendasi terkait jurnal dengan jumlah sebanyak 28 dengan topik yang relevan dengan artikel dari *input query* judul dan abstraknya. Penelitian selanjutnya bisa memperluas cakupan dengan menambah jenis jurnal yang dapat direkomendasikan.

DAFTAR PUSTAKA

Abadi, M.F. (2022) "Klasifikasi Lama Studi Mahasiswa Menggunakan Algoritma K-Nearest Neighbor," *Jurnal Teknologi Pintar*, 2(12), hal. 1–17.

Adimanggala, D. *dkk.*, (2021) "Evaluasi Topik Tersembunyi Berdasarkan Aspect Extraction menggunakan Pengembangan Latent Dirichlet Allocation," *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)*, 5(3), hal. 511–519. Tersedia pada: https://doi.org/10.29207/resti.v5i3.3075.

Badieah *dkk.*, (2023) "Exploiting Content-Based Filtering for Publication Venue Recommendations," (September), hal. 20–21.

Badriyah, T. *dkk.*, (2018) "Sistem Rekomendasi Content Based Filtering Menggunakan Algoritma Apriori," hal. 8–9.

Bhat, M.R. *dkk.*, (2020) "Deep LDA : A new way to topic model," *Journal of Information and Optimization Sciences*, 41(3), hal. 823–834. Tersedia pada: https://doi.org/10.1080/02522667.2019.1616911.

Cahyono *dkk.*, (2023) "Analisis Topic Modelling Persepsi Pengguna Internet Menggunakan Metode Latent Dirichlet Allocation," *Indonesian Journal of Computer Science*, 12(1), hal. 326–334. Tersedia pada: https://doi.org/10.33022/ijcs.v12i1.3155.

Dwi, Y. dan Maulana, F. (2021) "Perubahan Topik Media Tentang Program Keluarga Berencana Pasca Rebranding Bkkbn : Pendekatan Pemodelan Topik Digital," 6(02), hal. 10–20.

Khoury, H. El (2018) "Ontology-based Recommender System in Higher Education," 2, hal. 1031–1034. Tersedia pada: https://doi.org/10.1145/3184558.3191533.

Koda, A. *dkk.*, (2020) "Penentuan Bonus Karyawan Dengan Menggunakan Algoritma K-Nearest Neighbor," *KOPERTIP : Jurnal Ilmiah Manajemen Informatika dan Komputer*, 4(1), hal. 15–24. Tersedia pada: https://doi.org/10.32485/kopertip.v4i1.115.

Kurniawan *dkk.*, (2022) "Penerapan Text Mining Pada Sistem Penyeleksian Judul Skripsi Menggunakan Algoritma Latent Dirichlet Allocation(LDA),"

Indonesian Journal of Computer Science, 11(3), hal. 1036–1052. Tersedia pada: https://doi.org/10.33022/ijcs.v11i3.3120.

Larasati *dkk.*, (2021) "Sistem Rekomendasi Product Emina Cosmetics Dengan Menggunakan Metode Content - Based Filtering," *Jurnal Manajemen Informatika dan Sistem Informasi*, 4(1), hal. 45. Tersedia pada: https://doi.org/10.36595/misi.v4i1.250.

Lestandy *dkk.*, (2021) "Analisis Sentimen Tweet Vaksin COVID-19 Menggunakan Recurrent Neural Network dan Naïve Bayes," *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)*, 5(4), hal. 802–808. Tersedia pada: https://doi.org/10.29207/resti.v5i4.3308.

Manullang *dkk.*, (2021) "Penerapan Algoritma K-Nearest Neighbour Untuk Memprediksi Kelulusan Mahasiswa," *JIKOMSI Jurnal Ilmu Komputer dan Sistem Informasi*, 4(2), hal. 42–50.

Mouhoub, M. dan Al Helal, M. (2018) "Topic Modelling in Bangla Language: An LDA Approach to Optimize Topics and News Classification," *Computer and Information Science*, 11(4), hal. 77. Tersedia pada: https://doi.org/10.5539/cis.v11n4p77.

Munawar, Z. *dkk.*, (2021) "Dampak Intelijen Bisnis Pada Kualitas Pengambilan Keputusan," *Infotronik : Jurnal Teknologi Informasi dan Elektronika*, 6(1), hal. 32. Tersedia pada: https://doi.org/10.32897/infotronik.2021.6.1.661.

Nasrullah, D. (2023) "Klasifikasi Bidang Fokus Publikasi Jurnal."

Novarian, N., Khomsah, S. dan Arifa, A.B. (2023) "Topic Modeling Tugas Akhir Mahasiswa Fakultas Informatika Institut Teknologi Telkom Purwokerto Menggunakan Metode Latent Dirichlet Allocation Nathanael," *LEDGER: Journal Informatic and Information Technology*, 2(1), hal. 1–14.

Oktafiandi, H. (2023) "Implementasi LDA untuk Pengelompokan Topik Twitter Bertagar# Mypertamina," *Jurnal Ekonomi dan Teknik Informatika*, 11(1), hal. 10–16.

Purwitasari, D. *dkk.*, (2021) "Pemodelan Topik dengan LDA untuk Temu Kembali Informasi dalam Rekomendasi Tugas Akhir," *JURNAL RESTI (Rekayasa Sistem dan Teknologi Informasi)*, 5(3), hal. 421–428.

Rabbani *dkk.*, (2023) “Comparative Evaluation of SVM Kernels for Sentiment Classification in Fuel Price Increase Analysis Perbandingan Evaluasi Kernel SVM untuk Klasifikasi Sentimen dalam Analisis Kenaikan Harga BBM,” *MALCOM: Indonesian Journal of Machine Learning and Computer Science*, 3(2), hal. 153–160.

Rahayu *dkk.*, (2023) “Text Classification for Detecting Depression and Anxiety among Twitter Users based on Machine Learning Klasifikasi Teks untuk Mendeteksi Depresi dan Kecemasan Pada Pengguna Twitter Berbasis Machine Learning,” *MALCOM: Indonesian Journal of Machine Learning and Computer Science*, 3(2), hal. 108–114.

Ristanti, P. *dkk.*, (2019) “Cosine Similarity for Title and Abstract of Economic Journal Classification,” *Proceeding - 2019 5th International Conference on Science in Information Technology: Embracing Industry 4.0: Towards Innovation in Cyber Physical System, ICSITech 2019*, (October 2019), hal. 123–127. Tersedia pada: https://doi.org/10.1109/ICSITech46713.2019.8987547.

Rodrigues, O. (2018) “Combining Minkowski and Cheyshev: New distance proposal and survey of distance metrics using k-nearest neighbours classifier,” *Pattern Recognition Letters*, 110, hal. 66–71. Tersedia pada: https://doi.org/10.1016/j.patrec.2018.03.021.

Sa’adah, F. (2022) “Klasifikasi Bidang Ilmu Pada Publikasi Terindeks Garuda Menggunakan Metode K-Nearest Neighbor (K-Nn),” 5(2). Tersedia pada: http://repository.unissula.ac.id/id/eprint/29663%0Ahttp://repository.unissula.ac.id/29663/1/Teknik Informatika_32601800010_fullpdf.pdf.

Sanjaya, N.A. (2021) “lmplementasi Latent Dirichlet Allocation (LDA) untuk Klasterisasi Cerita Berbahasa Bali,” *Jurnal Teknologi Informasi dan Ilmu Komputer*, 8(1), hal. 127–134. Tersedia pada: https://doi.org/10.25126/jtiik.202183556.

Sari, N. *dkk.*, (2022) “Topic Modeling Using the Latent Dirichlet Allocation Method on Wikipedia Pandemic Covid-19 Data in Indonesia,” *Jurnal Teknik Informatika (Jutif)*, 3(5), hal. 1223–1230. Tersedia pada:

https://doi.org/10.20884/1.jutif.2022.3.5.321.

Sebastian, D. (2019) “Implementasi Algoritma K-Nearest Neighbor untuk Melakukan Klasifikasi Produk dari beberapa E-marketplace,” *Jurnal Teknik Informatika dan Sistem Informasi*, 5(1), hal. 51–61. Tersedia pada: https://doi.org/10.28932/jutisi.v5i1.1581.

Soyusiawaty, D. dan Zakaria, Y. (2018) “Book data content similarity detector with cosine similarity (case study on digilib.uad.ac.id),” *Proceeding of 2018 12th International Conference on Telecommunication Systems, Services, and Applications, TSSA 2018*, hal. 1–6. Tersedia pada: https://doi.org/10.1109/TSSA.2018.8708758.

Taunk, K. *dkk.,* (2019) “A brief review of nearest neighbor algorithm for learning and classification,” *2019 International Conference on Intelligent Computing and Control Systems, ICCS 2019*, (Iciccs), hal. 1255–1260. Tersedia pada: https://doi.org/10.1109/ICCS45141.2019.9065747.

Wijaya, K.H. (2022) *Final Project Classification of Science Fields in Garuda Indexed Publications Using the Naïve Bayes Method*.

Lampiran