

**PENERAPAN METODE LABELED LATENT DIRICHLET  
ALLOCATION DENGAN TF-IDF UNTUK KATEGORISASI  
DOKUMEN**

**TUGAS AKHIR**

Diajukan sebagai syarat untuk menyelesaikan  
Program Studi Strata-1 Departemen Informatika

**Disusun Oleh:**

**DEVIT LIE**

**1114001**



**INSTITUT  
TEKNOLOGI  
HARAPAN  
BANGSA**

*Veritas vos liberabit*

**DEPARTEMEN INFORMATIKA  
INSTITUT TEKNOLOGI HARAPAN BANGSA  
BANDUNG  
2017**



---

## LEMBAR PENGESAHAN

### PENERAPAN METODE LABELED LATENT DIRICHLET ALLOCATION DENGAN TF-IDF UNTUK KATEGORISASI DOKUMEN



Disusun oleh:  
Nama: Devit Lie  
NIM : 1114001

Telah Disetujui dan Disahkan  
Sebagai laporan Tugas Akhir Departemen Informatika  
Institut Teknologi Harapan Bangsa

Bandung, Desember 2017

Disetujui,

Pembimbing

Ria Chaniago, S.T., M.T.

**NIK. 114001**



### PERNYATAAN HASIL KARYA PRIBADI

Saya yang bertanda tangan di bawah ini:

Nama : Devit Lie

NIM : 1114001

Dengan ini menyatakan bahwa laporan Tugas Akhir dengan Judul : "**Penerapan Metode Labeled Latent Dirichlet Allocation Dengan TF-IDF Untuk Kategorisasi Dokumen**" adalah hasil pekerjaan saya dan seluruh ide, pendapat atau materi dari sumber lain telah dikutip dengan cara penulisan referensi yang sesuai.

Pernyataan ini saya buat dengan sebenar-benarnya dan jika pernyataan ini tidak sesuai dengan kenyataan maka saya bersedia menanggung sanksi yang akan dikenakan pada saya.

Bandung, Desember 2017

Yang membuat pernyataan,

Devit Lie

## **ABSTRAK**

Kategorisasi teks secara otomatis adalah salah satu solusi untuk membantu para pembaca menemukan data yang mereka inginkan dengan cepat. Dalam penelitian ini, metode yang digunakan untuk pembelajaran dan pengambilan keputusan adalah *Simplified Labeled Latent Dirichlet Allocation Classifier* (SLLDA-C) yang didasarkan pada metode *Labeled Latent Dirichlet Allocation* dan metode yang digunakan untuk seleksi fitur adalah TF-IDF. Langkah pertama adalah isi setiap dokumen yang telah diberi label akan dilakukan *pre-processing* yang meliputi proses *case folding*, *filtering*, *tokenizing*, *stopword removing*, dan *stemming*. Label yang digunakan dalam penitilian ini di antaranya adalah *machine learning*, *natural language processing*, dan *image processing*. Langkah selanjutnya adalah melakukan seleksi fitur terhadap isi dokumen dengan TF-IDF untuk mengurangi jumlah kata yang akan diolah ketika proses klasifikasi berlangsung. Kata-kata yang telah terpilih akan dilakukan pembelajaran dengan SLLDA-C untuk menghasilkan suatu *classifier model*. Pengujian akan dilakukan dengan jumlah data belajar sebanyak 180 data dan data uji sebanyak 20 data. Berdasarkan pengujian, dengan menggunakan TF-IDF hasil rata-rata *F-Measure* terbaik yang didapat adalah sebesar 90% dengan waktu pembelajaran yang dibutuhkan adalah selama 27 detik.

**Kata Kunci:** Kategorisasi Teks, SLLDA-C, *Labeled Latent Dirichlet Allocation*, TF-IDF, *F-Measure*

## ABSTRACT

*Automatic text categorization is one of the solution to help readers find the data they wanted quickly. In this research, the method used for learning and inference is Simplified Labeled Latent Dirichlet Allocation Classifier (SLLDA-C) based on Labeled Latent Dirichlet Allocation method and the method used for feature selection is TF-IDF. The first step is the contents of each document that has been labeled will be pre-processing which includes case folding, filtering, tokenizing, stopword removing, and stemming. The labels used in this research are machine learning, natural language processing, and image processing. The next step is to perform feature selection against the contents with TF-IDF for reducing the number of words that will be processed when learning process takes place. The words that have been selected will be learning with SLLDA-C to produce classifier model. The experiment will be done with 180 training datas and 20 testing datas. Based on the experiment, using TF-IDF the best average F-Measure results obtained is 90% with learning time for 27 seconds.*

**Keyword:** *Text Categorization, SLLDA-C, Labeled Latent Dirichlet Allocation, TF-IDF, F-Measure*

## **PEDOMAN PENGGUNAAN TUGAS AKHIR**

Laporan tugas akhir yang tidak dipublikasikan terdaftar dan tersedia di Perpustakaan Institut Teknologi Harapan Bangsa, dan terbuka untuk umum dengan ketentuan bahwa hak cipta ada pada pengarang dan pembimbing Tugas Akhir. Referensi kepustakaan diperkenankan dicatat, tetapi pengutipan atau peringkasan hanya dapat dilakukan dengan seizin pengarang atau pembimbing Tugas Akhir dan harus disertai dengan ketentuan penulisan ilmiah untuk menyebutkan sumbernya.

Tidak diperkenankan untuk memperbanyak atau menerbitkan sebagian atau seluruh laporan tugas akhir tanpa seizin dari pengarang atau pembimbing Tugas Akhir yang bersangkutan.

## KATA PENGANTAR

Terima kasih kepada Tuhan yang Maha Esa karena dengan bimbingan-Nya dan karunia-Nya penulis dapat melaksanakan Tugas Akhir yang berjudul "PENERAPAN METODE LABLED LATENT DIRICHLET ALLOCATION DENGAN TF-IDF UNTUK KATEGORISASI DOKUMEN". Laporan ini disusun sebagai salah satu syarat kelulusan di Institut Teknologi Harapan Bangsa. Pada kesempatan ini penulis menyampaikan terima kasih yang sebesar-besarnya kepada:

1. Tuhan Yang Maha Esa, karena oleh bimbingan-Nya penulis selalu mendapat pengharapan untuk menyelesaikan tugas akhir ini.
2. Ibu Ria Chaniago, S.T., M.T., selaku pembimbing I Tugas Akhir yang senantiasa memberi dukungan, semangat, ilmu-ilmu, saran dan dukungan kepada penulis selama tugas akhir berlangsung dan selama pembuatan laporan tugas akhir ini.
3. Bapak Firhat Hidayat, S.T., M.T., selaku penguji I Tugas Akhir. Terima kasih atas dukungan, semangat, ilmu-ilmu, dan masukan yang telah diberikan kepada penulis dalam menyelesaikan Laporan Tugas Akhir ini
4. Ibu Ir. Inge Martina, M.T., selaku penguji II dalam Tugas Akhir Terima kasih atas dukungan, semangat, ilmu-ilmu, dan masukan yang telah diberikan kepada penulis dalam menyelesaikan Laporan Tugas Akhir ini.
5. Seluruh dosen dan staff Departemen Teknik Informatika ITHB yang telah membantu dalam menyelesaikan Laporan Tugas Akhir ini.
6. Segenap jajaran staf dan karyawan ITHB yang turut membantu kelancaran dalam menyelesaikan Laporan Tugas Akhir ini.
7. Kedua orang tua tercinta yang selalu menyediakan waktu untuk memberikan doa, semangat dan dukungan yang tak habis-habisnya kepada penulis untuk menyelesaikan Laporan Tugas Akhir ini. Terima kasih untuk nasihat, masukan, perhatian, teguran dan

kasih sayang yang diberikan hingga saat ini.

Penulis menyadari bahwa laporan ini masih jauh dari sempurna karena keterbatasan waktu dan pengetahuan yang dimiliki oleh penulis. Oleh karena itu, kritik dan saran untuk membangun kesempurnaan tugas akhir ini sangat diharapkan. Semoga tugas akhir ini dapat membantu pihak-pihak yang membutuhkannya.

Bandung, Desemeber 2017

Hormat Saya,

Devit Lie.

## DAFTAR ISI

<b>LEMBAR PENGESAHAN</b>	<b>i</b>
<b>LEMBAR PERNYATAAN HASIL KARYA PRIBADI</b>	<b>ii</b>
<b>ABSTRAK</b>	<b>iii</b>
<b>ABSTRACT</b>	<b>iv</b>
<b>PEDOMAN PENGGUNAAN TUGAS AKHIR</b>	<b>v</b>
<b>KATA PENGANTAR</b>	<b>vi</b>
<b>DAFTAR ISI</b>	<b>x</b>
<b>DAFTAR TABEL</b>	<b>xiii</b>
<b>DAFTAR GAMBAR</b>	<b>xiv</b>
<b>I PENDAHULUAN</b>	<b>1-1</b>
1.1 Latar Belakang . . . . .	1-1
1.2 Rumusan Masalah . . . . .	1-2
1.3 Tujuan Penelitian . . . . .	1-2
1.4 Batasan Masalah . . . . .	1-2
1.5 Metode Penelitian . . . . .	1-3
1.6 Kontribusi Penelitian . . . . .	1-3
1.7 Sistematika Penulisan . . . . .	1-3
<b>II LANDASAN TEORI</b>	<b>2-1</b>
2.1 Tinjauan Pustaka . . . . .	2-1
2.1.1 Pemrosesan Bahasa Alami . . . . .	2-1
2.1.2 Klasifikasi Teks . . . . .	2-1
2.1.3 Pembelajaran Mesin . . . . .	2-1

2.1.3.1	Mesin Pembelajaran Terarah ( <i>Supervised Learning</i> ) . . . . .	2-2
2.1.4	<i>Text Pre-Processing</i> . . . . .	2-3
2.1.5	<i>Regular Expression</i> . . . . .	2-4
2.1.6	Algoritma <i>Snowball Stememer</i> . . . . .	2-4
2.1.7	Proses TF-IDF . . . . .	2-8
2.1.8	<i>Latent Dirichlet Allocation</i> . . . . .	2-10
2.1.9	<i>Labeled Latent Dirichlet Allocation</i> . . . . .	2-12
2.1.10	<i>Simplified Labeled Latent Dirichlet Allocation Classifier (SLLDA-C)</i> .	2-14
2.1.11	<i>Gibbs Sampling</i> . . . . .	2-15
2.1.12	<i>F-Measure</i> . . . . .	2-17
2.2	Tinjauan Studi . . . . .	2-19
2.2.1	<i>State of the Art</i> . . . . .	2-19
2.3	Tinjauan Objek . . . . .	2-19

<b>III ANALISIS DAN PERANCANGAN</b>	<b>3-1</b>	
3.1	Analisis Masalah . . . . .	3-1
3.2	Kerangka Pemikiran . . . . .	3-2
3.3	Perancangan . . . . .	3-3
3.3.1	<i>Data Sampling</i> . . . . .	3-4
3.3.2	<i>Pre-Processing</i> . . . . .	3-5
3.3.2.1	<i>Case Folding</i> . . . . .	3-5
3.3.2.2	<i>Filtering</i> . . . . .	3-6
3.3.2.3	<i>Tokenizing</i> . . . . .	3-6
3.3.2.4	<i>Stopword Removing</i> . . . . .	3-7
3.3.2.5	<i>Stemming</i> . . . . .	3-7
3.3.3	<i>Feature Selection</i> . . . . .	3-8
3.3.3.1	<i>Term Frequency</i> . . . . .	3-8
3.3.3.2	<i>Document Frequency</i> . . . . .	3-9
3.3.3.3	<i>Inverse Document Frequency</i> . . . . .	3-10
3.3.3.4	TF-IDF . . . . .	3-11
3.3.4	<i>Learning Process</i> . . . . .	3-12

3.3.5	<i>Classification</i>	3-22
3.4	Tahap Perancangan Implementasi Sistem	3-23
3.4.1	<i>Use Case Diagram</i>	3-23
3.4.2	<i>Class Diagram</i>	3-24
<b>IV IMPLEMENTASI DAN PENGUJIAN</b>		<b>4-1</b>
4.1	Lingkungan Implementasi	4-1
4.1.1	Spesifikasi Perangkat Keras	4-1
4.1.2	Lingkungan Perangkat Lunak	4-1
4.2	Implementasi Perangkat Lunak	4-1
4.2.1	Implementasi Basis Data	4-1
4.2.2	Daftar <i>Class dan Method</i>	4-3
4.2.3	Implementasi Struktur Data	4-10
4.3	Pengujian	4-13
4.3.1	Hasil Pengujian	4-13
4.3.1.1	Pengujian <i>Unigram</i> Tanpa TF-IDF dengan n TF-IDF	4-13
4.3.1.2	Pengujian <i>Bag of Words</i> Tanpa TF-IDF dengan n TF-IDF	4-16
4.3.2	Pengujian Waktu	4-18
4.3.3	Evaluasi	4-18
<b>V KESIMPULAN DAN SARAN</b>		<b>5-1</b>
5.1	Kesimpulan	5-1
5.2	Saran	5-2
<b>DAFTAR PUSTAKA</b>		<b>xv</b>

## DAFTAR TABEL

2.1	Langkah ke-2 <i>Snowball Stemmer</i> . . . . .	2-4
2.2	Langkah ke-2 <i>Snowball Stemmer</i> . . . . .	2-5
2.3	Kondisi ED dan ING Pada <i>Snowball Stemmer</i> . . . . .	2-5
2.4	Langkah ke-3 <i>Snowball Stemmer</i> . . . . .	2-5
2.5	Langkah ke-4 <i>Snowball Stemmer</i> . . . . .	2-6
2.6	Langkah ke-5 <i>Snowball Stemmer</i> . . . . .	2-6
2.7	Langkah ke-6 <i>Snowball Stemmer</i> . . . . .	2-7
2.8	<i>Term Count</i> . . . . .	2-8
2.9	<i>Term Frequency</i> . . . . .	2-8
2.10	<i>Document Frequency</i> . . . . .	2-9
2.11	<i>Inverse Document Frequency</i> . . . . .	2-9
2.12	TF-IDF . . . . .	2-10
2.13	Deskripsi Model LDA . . . . .	2-11
2.14	Deskripsi Model LLDA . . . . .	2-13
2.15	<i>Confusion Matrix</i> . . . . .	2-18
2.16	Contoh Hasil Perhitungan <i>F-Measure</i> . . . . .	2-18
2.17	<i>State of the Art</i> . . . . .	2-19
3.1	Contoh Data dalam <i>File .xls</i> . . . . .	3-4
3.2	Jumlah Data <i>Training</i> dan <i>Testing</i> . . . . .	3-4
3.3	Hasil <i>Case Folding</i> . . . . .	3-6
3.4	Hasil <i>Filtering</i> . . . . .	3-6
3.5	Hasil <i>Tokenizing</i> . . . . .	3-7
3.6	Hasil <i>Stopword Removing</i> . . . . .	3-7
3.7	Hasil <i>Stemming</i> . . . . .	3-8
3.8	Contoh <i>Term Count</i> . . . . .	3-9
3.9	Contoh <i>Term Frequency</i> . . . . .	3-9
3.10	Contoh Informasi Kata yang Muncul . . . . .	3-10

3.11 Contoh <i>Document Frequency</i> . . . . .	3-10
3.12 Contoh <i>Inverse Document Frequency</i> . . . . .	3-11
3.13 Contoh TF-IDF . . . . .	3-11
3.14 Contoh Isi Dokumen Beserta Label . . . . .	3-12
3.15 Contoh Nilai Masukan LLDA . . . . .	3-13
3.16 Contoh Pemetaan Ada Tidaknya Suatu Topik Pada Setiap Dokumen . . . . .	3-13
3.17 Contoh Hasil Pengindeksan Terhadap Semua Kata . . . . .	3-13
3.18 Contoh Matriks Awal Jumlah Topik Terhadap Kata . . . . .	3-14
3.19 Contoh Matriks Awal Jumlah Dokumen Terhadap Topik . . . . .	3-14
3.20 Contoh Awal Penetapan Topik Terhadap Semua Kata . . . . .	3-14
3.21 Contoh Hasil Random Topic Untuk Semua Kata . . . . .	3-15
3.22 Contoh Hasil Penambahan Nilai Matriks <i>Topic-Word</i> . . . . .	3-15
3.23 Contoh Hasil Penambahan Nilai Matriks <i>Document-Topic</i> . . . . .	3-16
3.24 Contoh Pengamsumsian Topik Untuk Kata ke-1 Dokumen 1 . . . . .	3-16
3.25 Contoh Pengurangan Nilai Matriks <i>Topic-Word</i> Untuk Kata ke-1 Dokumen 1 . . . . .	3-17
3.26 Contoh Pengurangan Nilai Matriks <i>Document-Topic</i> Untuk Topik ke-2 Dokumen 1 . . . . .	3-17
3.27 Contoh Perhitungan Probabilitas <i>Topic-Word</i> dan <i>Document-Topic</i> . . . . .	3-17
3.28 Contoh Perhitungan Distribusi Untuk Semua Topik . . . . .	3-18
3.29 Contoh Perhitungan Normalisasi Distribusi Untuk Semua Topik . . . . .	3-18
3.30 Contoh Perhitungan Akumulasi Nilai Normalisasi Distribusi Untuk Semua Topik	3-19
3.31 Contoh Proses Perbandingan Nilai Distribusi Setiap Topik . . . . .	3-19
3.32 Contoh Penetapan Kembali Topik Untuk Kata ke-1 Dokumen 1 . . . . .	3-20
3.33 Contoh Pengamsumsian Topik Untuk Kata ke-1 Dokumen 1 . . . . .	3-20
3.34 Contoh Penambahan Kembali Jumlah Kata Matriks <i>Document-Topic</i> . . . . .	3-20
3.35 Contoh Hasil Matriks <i>Topic-Word</i> Dikalikan dengan $\beta$ . . . . .	3-21
3.36 Contoh Proses Perbandingan Nilai Distribusi Setiap Topik . . . . .	3-21
3.37 Contoh Hasil Perhitungan Nilai $\phi$ . . . . .	3-21
3.38 Contoh 20% Kata Dengan Probabilitas Tertinggi Untuk Setiap Topik . . . . .	3-22
3.39 Contoh Pemilihan Kategori dengan Menggunakan Nilai <i>Threshold</i> . . . . .	3-23

4.1	Struktur Tabel Data_Training . . . . .	4-2
4.2	Struktur Tabel Data_Testing . . . . .	4-2
4.3	Struktur Tabel LLDA . . . . .	4-2
4.4	Daftar <i>Class</i> pada <i>Package final_project_api</i> . . . . .	4-3
4.5	Daftar <i>Class</i> pada <i>Package final_project_impl</i> . . . . .	4-4
4.6	Daftar Atribut <i>Dataset Specification</i> . . . . .	4-5
4.7	Daftar Fungsi dan Prosedur <i>Dataset Specification</i> . . . . .	4-5
4.8	Daftar Atribut <i>LLDA Specification</i> . . . . .	4-6
4.9	Daftar Fungsi dan Prosedur <i>LLDA Specification</i> . . . . .	4-6
4.10	Daftar Fungsi dan Prosedur <i>Dataset Accessor</i> . . . . .	4-6
4.11	Daftar Fungsi dan Prosedur <i>Dataset Service</i> . . . . .	4-7
4.12	Daftar Fungsi dan Prosedur <i>Pre-Processing Service</i> . . . . .	4-7
4.13	Daftar Fungsi dan Prosedur <i>Feature Selection Service</i> . . . . .	4-8
4.14	Daftar Fungsi dan Prosedur <i>Classifier Service</i> . . . . .	4-8
4.15	Daftar Fungsi dan Prosedur <i>LLDA Accessor</i> . . . . .	4-9
4.16	Daftar Fungsi dan Prosedur <i>Learning Service</i> . . . . .	4-9
4.17	Daftar Fungsi dan Prosedur <i>Inference Service</i> . . . . .	4-9
4.18	Daftar Fungsi dan Prosedur <i>Measurement Service</i> . . . . .	4-10
4.19	Parameter Pengujian . . . . .	4-13
4.20	Pengujian <i>Unigram</i> Tanpa TF-IDF <i>Single Label</i> . . . . .	4-13
4.21	Pengujian <i>Unigram</i> n TF-IDF <i>Single Label</i> . . . . .	4-14
4.22	<i>Confusion Matrix Unigram</i> Tanpa TF-IDF <i>Single Label</i> . . . . .	4-15
4.23	<i>Confusion Matrix Unigram</i> n TF-IDF <i>Single Label</i> . . . . .	4-15
4.24	Pengujian <i>Unigram</i> tanpa TF-IDF <i>Multi Label</i> . . . . .	4-15
4.25	Pengujian <i>Unigram</i> n TF-IDF <i>Multi Label</i> . . . . .	4-16
4.26	Pengujian <i>Bigram</i> dan <i>Trigram</i> . . . . .	4-17
4.27	Pengujian Waktu Proses Pembelajaran . . . . .	4-18
4.28	Contoh Permasalahan Menggunakan TF-IDF . . . . .	4-19
4.29	<i>Confusion Matrix Unigram</i> Terbaik . . . . .	4-19
4.30	<i>Confusion Matrix Bigram</i> Terbaik . . . . .	4-20

## **DAFTAR GAMBAR**

2.1	<i>Model LDA</i>	2-10
2.2	<i>Model LLDA</i>	2-12
3.1	Kerangka Pemikiran LLDA	3-3
3.2	<i>Flowchart Global</i>	3-3
3.3	<i>Flowchart Pre-Processing</i>	3-5
3.4	<i>Flowchart Feature Selection</i>	3-8
3.5	<i>Use Case Diagram</i>	3-24
3.6	<i>Class Diagram</i>	3-24

# BAB I

## PENDAHULUAN

### 1.1 Latar Belakang

Seiring dengan pertumbuhan internet yang semakin cepat, ketersediaan dari informasi teks juga ikut meningkat pesat. Terkadang para pembaca membutuhkan waktu untuk menemukan data yang mereka inginkan. Salah satu solusi untuk menangani masalah tersebut adalah dengan menggunakan kategorisasi teks otomatis. Penelitian ini akan menciptakan sebuah sistem kategorisasi yang dapat mengelompokkan sejumlah data ke dalam beberapa kelompok yang sesuai. Data yang akan digunakan pada penelitian ini adalah data mengenai dokumen penelitian. Dari kumpulan data tersebut, data akan dikelompokkan ke dalam beberapa kelompok berdasarkan topik yang sesuai. Bagian dokumen penelitian yang akan digunakan sebagai masukan untuk sistem kategorisasi adalah bagian abstrak.

Beberapa metode yang umum digunakan untuk kategorisasi di antaranya adalah *Support Vector Machine* [1], K-NN [2], dan *Naive Bayes* [3]. *Support Vector Machine* memiliki kelebihan untuk mengatasi beberapa fitur yang tidak relevan, teks yang jarang muncul, dan dapat melakukan generalisasi *sample* dengan baik (menghindari *overfitting*) tetapi SVM memiliki kelemahan juga pada sulitnya pemilihan parameter SVM yang optimal [1]. Metode K-NN efektif jika sistem memiliki data *training* dalam jumlah besar. Dengan K-NN juga, data *training robust* terhadap data *noise*, tetapi K-NN memiliki kelemahan pada penentuan nilai K dan atribut yang digunakan (tidak efisien) serta biaya komputasinya tinggi [2]. *Naïve Bayes* memiliki kelebihan yaitu algoritma yang digunakan sederhana, tetapi *Naïve Bayes* memiliki kelemahan untuk menangani data yang jarang muncul dan penentuan parameter yang bersifat independen [3].

Metode yang telah disebutkan di atas hanya mengelompokkan dokumen ke dalam 1 kategori yang paling dominan saja (*single label*), tetapi ada keadaan dimana suatu dokumen dapat memiliki lebih dari 1 kategori (*multi label*). *Labeled Latent Dirichlet Allocation* merupakan salah satu metode yang memungkinkan sebuah dokumen untuk memiliki lebih dari 1 kategori. Untuk menggunakan metode *Labeled Latent Dirichlet Allocation*, ada beberapa hal yang perlu dilakukan adalah melakukan *pre-processing* terhadap kata-kata yang terdapat pada dokumen, menghitung nilai TF-IDF, dan melakukan pembelajaran atau menghasilkan kesimpulan dengan menggunakan *Gibbs Sampling*. Kelebihan dari metode ini adalah dapat menghasilkan lebih dari 1 kategori untuk setiap dokumen, memecahkan masalah dari K-NN (tidak efisien dan waktu komputasi tinggi) dan *Naïve Bayes* (tidak fleksibel), serta dapat mengungguli kelebihan yang dimiliki SVM juga.

Meskipun *Labeled Latent Dirichlet Allocation* dapat dikatakan sebagai salah satu

metode klasifikasi yang baik saat ini, tingkat akurasi dan waktu komputasi dari metode *Labeled Latent Dirichlet Allocation* masih dapat ditingkatkan dan dikurangi lagi untuk dapat menghasilkan hasil yang lebih akurat dan cepat [4]. Berdasarkan hal tersebut, salah satu pendekatan yang dapat digunakan adalah menggunakan metode *feature selection*, dimana metode ini berfungsi untuk mendapatkan kata-kata yang penting saja. Metode *feature selection* yang digunakan adalah TF-IDF, sehingga metode yang akan digunakan dalam penelitian ini adalah penggabungan *Labeled Latent Dirichlet Allocation* dan TF-IDF. Hasil dari sistem kategorisasi ini adalah berupa kategori (label).

### 1.2 Rumusan Masalah

Berdasarkan masalah yang telah diuraikan di atas, maka dapat dirumuskan beberapa masalah yang di antaranya adalah :

1. Bagaimana menerapkan *Labeled Latent Dirichlet Allocation* untuk melakukan pengelompokan dokumen penelitian?
2. Bagaimana proses kategorisasi dokumen penelitian dengan *Labeled Latent Dirichlet Allocation*?
3. Bagaimana meningkatkan tingkat akurasi serta menurunkan waktu komputasi proses kategorisasi dokumen penelitian menggunakan *Labeled Latent Dirichlet Allocation*?
4. Apa pengaruh TF-IDF terhadap hasil akhir *Labeled Latent Dirichlet Allocation*?

### 1.3 Tujuan Penelitian

Berdasarkan latar belakang dan rumusan masalah tersebut, maka tujuan utama dari penelitian ini adalah :

1. Mengelompokan dokumen penelitian ke dalam beberapa kategori untuk dijadikan sebagai *corpus*.
2. Dapat menentukan beberapa kategori yang sesuai terhadap dokumen penelitian yang diuji.
3. Meneliti bahwa dengan adanya metode *feature selection* pada proses *Labeled Latent Dirichlet Allocation*, tingkat akurasi yang akan dihasilkan akan lebih akurat dan waktu komputasi yang dibutuhkan akan lebih cepat.

### 1.4 Batasan Masalah

Agar penelitian ini dapat sesuai dengan tujuan utama yang telah disebutkan di atas, maka digunakan beberapa batasan seperti di bawah ini :

1. Dokumen penelitian yang digunakan adalah dokumen berbahasa Inggris.
2. *Stemmer* yang akan digunakan untuk melakukan proses *stemming* adalah *Snowball Stemmer*.
3. Dokumen penelitian yang digunakan bersumber dari kumpulan tugas akhir mahasiswa Institut Teknologi Harapan Bangsa jurusan Informatika, IEEE, ACM, dan *Research Gate*.
4. Pada proses kategorisasi sistem tidak dapat menangani frase, homonim, homograf, polisemi, singkatan, dan kata tidak baku.

5. Bagian dokumen penelitian yang akan digunakan untuk pengolahan data adalah bagian abstrak.
6. Label yang digunakan pada dokumen penelitian adalah *Machine Learning*, *Natural Language Processing*, dan *Image Processing*.
7. Sistem yang dibuat hanya berbasis *web* dengan sistem operasi *Windows*.

### **1.5 Metode Penelitian**

Di bawah ini merupakan tahapan-tahapan yang akan dilakukan selama penelitian :

1. Studi literatur mengenai *text pre-processing*, *feature selection*, *text classification*, dan algoritma *Labeled Latent Dirichlet Allocation* yang digunakan untuk kategorisasi dokumen.
2. Analisis terhadap *Labeled Latent Dirichlet Allocation* serta algoritma-algoritma yang digunakan.
3. Perancangan sistem kategorisasi meliputi perancangan *database*, fungsi-fungsi yang terdapat dalam sistem, algoritma yang digunakan, dan perancangan antarmuka.
4. Implementasi algoritma *Labeled Latent Dirichlet Allocation* pada sistem kategorisasi berdasarkan rancangan yang telah dibuat.
5. Pengujian terhadap hasil penelitian dari segi ketepatan saat digunakan.

### **1.6 Kontribusi Penelitian**

Pada penelitian ini, penulis akan melakukan kategorisasi dokumen penelitian dengan menggunakan metode *Labeled Latent Dirichlet Allocation*. Penulis akan mencoba menggunakan bagian abstrak pada dokumen penelitian sebagai data yang akan diolah untuk pembelajaran maupun pengambilan keputusan. Selain itu juga, Penulis akan mencoba untuk mengimplementasikan metode *feature selection* pada penelitian ini. Metode *feature selection* yang akan penulis gunakan adalah TF-IDF. Penggunaan metode *feature selection* ini merupakan salah satu tujuan penulis untuk meneliti apakah dengan TF-IDF tingkat akurasi yang didapatkan akan semakin akurat serta apakah dengan TF-IDF juga waktu komputasi yang dibutuhkan mesin dapat lebih cepat dibanding tanpa menggunakan TF-IDF.

### **1.7 Sistematika Penulisan**

Penyusunan penelitian ini akan dibuat dalam beberapa tahapan seperti di bawah ini :

#### **BAB I PENDAHULUAN**

Bab ini merupakan bagian pendahuluan, yang menjelaskan tentang kenapa penulis ingin melakukan penelitian ini, rumusan masalah, tujuan utama dari penelitian, batasan masalah, metodologi penelitian, kontribusi penelitian, serta sistematika penulisan.

#### **BAB II LANDASAN TEORI**

Bab ini merupakan bagian landasan teori, yang menjelaskan tentang studi literatur

yang telah dilakukan lalu dituangkan sebagai landasan teori penelitian ini.

**BAB III ANALISIS DAN PERANCANGAN**

Bab ini merupakan bagian analisis dan perancangan, yang menjelaskan tentang proses analisa hingga perancangan dari penelitian ini.

**BAB IV IMPLEMENTASI DAN PENGUJIAN**

Bab ini merupakan bagian implementasi dan pengujian, yang menjelaskan implementasi sistem dan pengujian terhadap sistem yang telah dibangun.

**BAB V KESIMPULAN DAN SARAN**

Bab ini merupakan bagian penutup, yang menjelaskan kesimpulan dan saran dari penelitian yang telah dilakukan.

## BAB II

### LANDASAN TEORI

#### 2.1 Tinjauan Pustaka

Tinjauan pustaka merupakan bagian yang menjelaskan semua dasar teori yang dibutuhkan untuk melakukan penelitian ini.

##### 2.1.1 Pemrosesan Bahasa Alami

Pemrosesan Bahasa Alami atau yang biasa disebut dengan singkatan PBA atau NLP adalah cara bagi komputer untuk menganalisa, memahami, dan memperoleh makna dari bahasa manusia dengan cara yang cerdas dan berguna. Interaksi manusia komputer ini memungkinkan aplikasi dunia nyata seperti *automatic text summarization, sentiment analysis, topic extraction, named entity recognition, parts-of-speech tagging, relationship extraction, stemming*, dll [5].

NLP biasanya digunakan untuk *information retrieval, text mining, machine translation*, dan *automated question answering*. Jika dikaitkan dengan permasalahan pada penelitian ini, *text categorization* merupakan salah satu bagian terpenting dalam kasus *information retrieval* untuk meningkatkan akurasi informasi serta meningkatkan kepuasan pengguna terhadap informasi yang didapat [6].

##### 2.1.2 Klasifikasi Teks

Klasifikasi Teks atau Kategorisasi Teks merupakan salah satu masalah pada *machine learning* yang bertujuan untuk mengelompokkan sekumpulan dokumen ke dalam kelompok tertentu. Masukan pada klasifikasi teks adalah berupa dokumen-dokumen yang telah diberi label. Label yang terdapat pada setiap dokumen dapat berjumlah 1 atau lebih, karena setiap dokumen pasti memiliki lebih dari 1 kategori.

Dengan menggunakan *machine learning*, masukan yang telah disebutkan di atas akan digunakan sistem untuk belajar agar dapat mengenali pola dari dokumen yang diberikan, dimana pola yang didapat akan menghasilkan suatu model. Model ini kemudian akan digunakan sistem untuk menghasilkan sebuah keluaran berupa kategori atau label. Contoh sederhana untuk klasifikasi teks adalah *movie genre classification*, dimana masukan kasus ini dapat berupa sinopsis dari setiap film dan label yang akan digunakan untuk setiap film adalah “*thriller*”, “*romantic*”, “*terror*”, “*horror*”, dll.

##### 2.1.3 Pembelajaran Mesin

Pembelajaran Mesin (*machine learning*) merupakan cara bagi komputer untuk belajar agar dapat berpikir dan mengambil keputusan seperti manusia. Masukan yang dibutuhkan oleh komputer adalah data yang memiliki fitur yang sesuai dengan masalah yang ingin diselesaikan.

Sebagai contoh jika masalah yang dihadapi merupakan kategorisasi buku maka data yang dibutuhkan dapat berupa buku-buku dengan sinopsis sebagai fiturnya. Data ini akan digunakan komputer sebagai sumber pembelajaran dalam mengambil keputusan.

Pembelajaran mesin pada dunia nyata dapat diimplementasikan untuk *text classification*, *fraud detection*, *pattern and image recognition*, *new pricing models*, dll. Secara mendasar teknik pada pembelajaran mesin dapat diklasifikasikan menjadi 3 kelompok. Kelompok ini di antaranya adalah *supervised learning*, *unsupervised learning*, dan *reinforcement learning*. Pada penelitian ini, teknik pembelajaran mesin yang akan digunakan adalah *supervised learning* dan masalah yang akan diteliti adalah mengenai klasifikasi teks. Berikut merupakan beberapa terminologi yang terdapat dalam pembelajaran mesin [1].

1. Data Belajar, merupakan data yang digunakan selama proses pembelajaran untuk menghasilkan data model. Data pada data belajar harus berisi informasi yang sesuai.
2. Data Model, merupakan data yang didapat dari proses belajar yang berfungsi untuk basis dalam pengambilan keputusan.
3. Proses Belajar, merupakan proses mengolah data belajar agar dapat menghasilkan suatu model.
4. Data Uji, merupakan data yang digunakan ketika melakukan pengujian. Proses pengujian akan menggunakan model yang didapat dari proses belajar untuk menentukan kategori dari data yang ingin diuji.
5. Fitur (*Feature*), merupakan karakteristik unik yang dimiliki sebuah objek.
6. Seleksi fitur (*Feature Selection*) merupakan proses memilih *subset* dari fitur yang relevan untuk digunakan dalam pembangunan model.
7. *Classification*, merupakan proses mengambil beberapa jenis masukan dan menetapkan label pada masukan tersebut.

### 2.1.3.1 Mesin Pembelajaran Terarah (*Supervised Learning*)

Mesin Pembelajaran Terarah secara mendasar merupakan salah satu bagian dari teknik pembelajaran mesin untuk membuat komputer agar dapat berpikir seperti manusia. Mesin pembelajaran terarah melakukan pembelajaran terhadap mesin dengan bantuan manusia untuk menentukan suatu pilihan. Masukan dari teknik mesin pembelajaran terarah harus memiliki label pada setiap data belajar yang akan digunakan. Label yang ditetapkan dapat berupa *single label* (1 label) maupun *multi label* (lebih dari 1 label). Berikut merupakan langkah-langkah dalam memecahkan masalah pada mesin pembelajaran terarah.

1. Menentukan jenis data yang akan digunakan sebagai data belajar. Dalam kasus kategorisasi teks, contoh data yang dapat digunakan adalah abstraksi jurnal penelitian, isi berita, atau sinopsis film.
2. Kumpulkan data belajar, dimana data belajar harus merepresentasikan kasus pada dunia nyata.
3. Menentukan masukan fitur yang akan digunakan dalam proses belajar. Jumlah fitur dapat

mempengaruhi tingkat akurasi yang dihasilkan. Jumlah fitur tidak boleh terlalu besar, namun harus berisi cukup informasi untuk menghasilkan keluaran yang akurat.

4. Menentukan algoritma pembelajaran yang sesuai. Sebagai contoh, algoritma yang dapat digunakan untuk kategorisasi adalah *Labeled Latent Dirichlet Allocation*, *Support Vector Machine*, atau *Naïve Bayes*.
5. Melakukan proses pembelajaran terhadap data belajar menggunakan algoritma *supervised learning*. Beberapa algoritma *supervised learning* mengharuskan pengguna untuk menentukan parameter tertentu. Parameter ini digunakan untuk mengoptimalkan kinerja dari data belajar.
6. Mengevaluasi akurasi dengan cara menguji hasil data belajar yang telah dipelajari dengan data uji.

### 2.1.4 *Text Pre-Processing*

*Pre-processing* adalah proses normalisasi teks sehingga informasi yang dimuat merupakan bagian yang padat dan ringkas namun tetap merepresentasikan informasi yang termuat di dalamnya. Tahapan untuk melakukan *pre-processing* ini adalah *case folding*, *filtering*, *tokenizing*, *stopword removing*, dan *stemming* [7].

#### 1. *Case Folding*

*Case Folding* merupakan proses mengubah bentuk huruf pada setiap kata menjadi bentuk huruf kecil. Tahap ini berfungsi untuk mengurangi jumlah fitur kata yang digunakan.

#### 2. *Filtering*

*Filtering* merupakan proses untuk menghapus tanda baca dan kata penghubung pada setiap kalimat. Tahap ini berfungsi untuk mempermudah pembacaan setiap kata untuk proses selanjutnya.

#### 3. *Tokenizing*

*Tokenizing* merupakan proses pemotongan dari bentuk kalimat menjadi kumpulan kata-kata yang disebut sebagai *token*. *Token* akan digunakan sebagai representasi data pada tiap baris yang ada di matriks probabilitas. Teknik *tokenizing* untuk mengubah suatu kalimat menjadi sebuah *token* adalah *word tokenizing*. Sebagai contoh bila terdapat dokumen berisi kalimat sebagai berikut.

“I want to eat rice”

Maka hasil *word tokenizing* dari kalimat tersebut akan menjadi seperti di bawah ini.

[I] [want] [to] [eat] [rice]

#### 4. *Stopword Removing*

*Stopword Removing* merupakan proses penghapusan kata-kata yang tidak mewakili makna dari suatu kalimat. Salah satu sumber untuk mendapatkan daftar *stopword* adalah pada website [www.ranks.nl](http://www.ranks.nl). Contoh *stopword* yang didapat dari sumber tersebut di antaranya seperti “*am, is, are, and, in, none, however, the, etc*”.

#### 5. *Stemming*

*Stemming* merupakan proses untuk merubah sebuah kata menjadi kata dasar sesuai dengan

pembentuk kata tersebut. Caranya adalah dengan menghilangkan imbuhan yang melekat pada kata, sehingga hasilnya adalah kata dasarnya. Dalam Bahasa Inggris, proses *stemming* bisa mengikutsertakan pengembalian bentuk *tense* dari kata kerja bentuk ke-2 atau ke-3 menjadi kata kerja bentuk ke-1. Sebagai contoh dalam kamus Bahasa Inggris kata “*categorization*” berasal dari kata “*categorize*”.

### 2.1.5 Regular Expression

*Regular Expression (regex)* merupakan teks khusus yang digunakan sebagai pola untuk mencocokan sekumpulan *string*. *Regular expression* mulai muncul pada tahun 1940-an sebagai cara untuk menggambarkan bahasa-bahasa pada umumnya, namun *regular expression* benar-benar muncul pada dunia *programming* sekitar tahun 1970-an. *Regular expression* pertama kali digunakan pada QED *text editor* oleh Ken Thompson. Sebagai contoh *regex* untuk mencocokkan angka antara 0 sampai 9 adalah seperti di bawah ini [8].

$\sim(\backslash(\backslash d\{3\}\backslash)|^\backslash d\{3\}[-]?)?\backslash d\{3\}[-]?\backslash d\{4\}\$$

### 2.1.6 Algoritma Snowball Stememer

*Snowball Stemmer* merupakan metode *stemming* yang dikembangkan dari *Porter Stemmer*. Algoritma *Porter Stemmer* mulai diperkenalkan pada tahun 1980. Algoritma ini merupakan salah satu *stemmer* yang umum digunakan karena menghasilkan hasil *stemming* yang baik dibanding *stemmer* yang lainnya, selain itu juga nilai *error* yang dihasilkan *Porter Stemmer* dapat dikatakan kecil [9]. Untuk dapat lebih memahami algoritma dari *Porter Stemmer*, berikut merupakan cara kerja penggunaan algoritma *Snowball Stemmer* (*Porter Stemmer* yang dikembangkan).

1. Menghapus akhiran ‘, ‘s, dan ‘s’ jika ditemukan.
2. Menangani *plural* dan *past participle*. Mencari akhiran pada Tabel 2.1 dan mengganti akhiran tersebut sesuai dengan tugasnya masing-masing.

Tabel 2.1 Langkah ke-2 *Snowball Stemmer*

Akhiran	Hasil ( <i>Replace</i> )	Contoh
SSES	SS	Caresses → Caress
IES	I (Jika lebih dari 1 huruf)	Ponies → Poni
	IE	Ties → tie
SS	SS	Caress → Caress
S		Cats → Cat
EED	EE (Jika lebih dari 1 huruf)	Agreed → Agree
		Feed → Feed

## II. LANDASAN TEORI

---

**Tabel 2.2** Langkah ke-2 *Snowball Stemmer*

ED, ING		Motoring → Motor
Y	I (Jika didahului oleh konsonan yang bukan huruf pertama dari kata tersebut)	Cry → Cri

Terdapat penanganan khusus untuk akhiran ED dan ING setelah dilakukan penghapusan. Tabel 2.2 merupakan penanganan untuk akhiran ED dan ING.

**Tabel 2.3** Kondisi ED dan ING Pada *Snowball Stemmer*

Kondisi	Hasil ( <i>Replace</i> )	Contoh
Di akhiri at, bl, iz	e	Luxuriat → Luxuriate
Di akhiri 2 huruf yang sama		Hopp → Hop
Terdiri dari 3 huruf (short)	e	Hop → Hope

3. Mencari akhiran pada Tabel 2.3 dan mengganti akhiran tersebut sesuai dengan tugasnya masing-masing.

**Tabel 2.4** Langkah ke-3 *Snowball Stemmer*

Akhiran	Hasil ( <i>Replace</i> )	Contoh
ATIONAL	ATE	Relational → Relate
TIONAL	TION	Conditional → Condition
ENCI	ENCE	Valenci → Valence
ANCI	ANCE	Hesitanci → Hesitance
IZER	IZE	Digitizer → Digitize
ABLI	ABLE	Conformabli → Conformable
ALLI	AL	Radicalli → Radical
ENTLI	ENT	Differentli → Different
ELI	E	Vileli → Vile
OUSLI	OUS	Analogousli → Analogous
IZATION	IZE	Vietnamization → Vietnamize
ATION	ATE	Predication → Predicate

## II. LANDASAN TEORI

---

ATOR	ATE	Operator → Operate
ALISM	AL	Feudalism → Feudal
IVENESS	IVE	Decisiveness → Decisive
FULNESS	FUL	Hopefulness → Hopeful
OUSNESS	OUS	Callousness → Callous
ALITI	AL	Formaliti → Formal
IVITI	IVE	Sensitiviti → Sensitive
BILITI	BLE	Sensibiliti → Sensible

4. Mencari akhiran pada Tabel 2.4 dan mengganti akhiran tersebut sesuai dengan tugasnya masing-masing.

**Tabel 2.5** Langkah ke-4 Snowball Stemmer

Akhiran	Hasil ( <i>Replace</i> )	Contoh
ICATE	IC	TriPLICATE → triplic
ATIVE		FORMATIVE → form
ALIZE	AL	FORMALIZE → formal
ICITI	IC	ELECTRICITI → electric
ICAL	IC	ELECTRICAL → electric
FUL		HOPFUL → hope
NESS		GOODNESS → good

5. Mencari akhiran pada Tabel 2.5 dan mengganti akhiran tersebut sesuai dengan tugasnya masing-masing. Langkah ini merupakan langkah terakhir untuk menghapus akhiran dalam Bahasa Inggris.

**Tabel 2.6** Langkah ke-5 Snowball Stemmer

Akhiran	Hasil ( <i>Replace</i> )	Contoh
AL		REVIVAL → Reviv
ANCE		ALLOWANCE → Allow
ENCE		INFERENCE → Infer

## II. LANDASAN TEORI

---

ER		Airliner → Airlin
IC		Gyroscopic → Gyroscop
ABLE		Adjustable → Adjust
IBLE		Defensible → Defens
ANT		Irritant → Irrit
EMENT		Replacement → Replac
MENT		Adjustment → Adjust
ENT		Dependent → Depend
ION		Adoption → Adopt
OU		Homologlou → Homolog
ISM		Communism → Commun
ATE		Activate → Activ
ITI		Angulariti → Angular
OUS		Homologlous → Homolog
IVE		Effective → Effect
IZE		Bowdlerize → Bowdler

6. Merapikan kata-kata yang telah melalui tahap penghapusan akhiran. Tabel 2.6 merupakan contoh langkah untuk merapikan kata-kata.

**Tabel 2.7** Langkah ke-6 *Snowball Stemmer*

Akhiran	Hasil ( <i>Replace</i> )	Contoh
E		Probate → Probat
		Rate → Rate
		Cease → Ceas
L		Controll → Control
		Roll → Roll

### 2.1.7 Proses TF-IDF

TF-IDF merupakan salah satu metode *feature selection* yang digunakan untuk mencari bobot nilai setiap kata yang ada pada dokumen. Setiap kata yang ada pada setiap dokumen direpresentasikan dalam bentuk vektor. Tahapan proses yang dibutuhkan untuk menghitung TF-IDF adalah menghitung *term frequency*, *document frequency*, kemudian *inverse document frequency* [7].

#### 1. Term Frequency

*Term Frequency* (tf) merupakan frekuensi kemunculan *term* (t) pada dokumen (d). Pada tahap ini, matriks *term-document* akan dibangun dengan menempatkan kata hasil proses *stemming* (*term*) ke dalam baris. Setiap baris mewakili sebuah kata yang unik, sedangkan setiap kolom mewakili konteks dari mana kata-kata tersebut diambil. Konteks yang dimaksud bisa berupa kalimat, paragraf, atau seluruh bagian dari teks.

**Tabel 2.8 Term Count**

<b>Term (t)</b>	<b>Term Count</b>		
	Document 1	Document 2	Document 3
Natural	0	1	N
Recommend	1	2	N
System	1	1	N

$$TF = \frac{TC_w^{(d)}}{TC^{(d)}} \quad (2 . 1)$$

$TC_w^{(d)}$  = Jumlah Kata w Pada Dokumen d

$TC^{(d)}$  = Jumlah Seluruh Kata Pada Dokumen d

**Tabel 2.9 Term Frequency**

<b>Term(t)</b>	<b>Term Frequency</b>	
	<b>Document 1</b>	<b>Document 2</b>
Natural	$0/2 = 0$	$1/6 = 0.167$
Recommend	$1/2 = 0.5$	$2/6 = 0.333$
System	$1/2 = 0.5$	$3/6 = 0.5$

## 2. Document Frequency

*Document Frequency* (df) adalah banyaknya dokumen dimana suatu *term* (t) muncul. Nilai *document frequency* untuk setiap kata harus lebih kecil atau sama dengan jumlah dokumen yang tersedia. Tabel 2.9 merupakan contoh *document frequency*.

**Tabel 2.10 Document Frequency**

Term (t)	df
Natural	1
Recommend	2
System	2

## 3. Inverse Document Frequency

*Inverse Document Frequency* (idf) berfungsi mengurangi bobot suatu *term* jika kemunculannya banyak tersebar di seluruh koleksi dokumen. Persamaan 2.2 merupakan rumus untuk menghitung nilai *inverse document frequency*.

$$idf = \log\left(\frac{N}{df}\right) \quad (2.2)$$

*N* = Jumlah Dokumen

*df* = Document Frequency

**Tabel 2.11 Inverse Document Frequency**

Term (t)	df	idf
Natural	1	$\log(2/1) = 0.301$
Recommend	2	$\log(2/2) = 0$
System	2	$\log(2/2) = 0$

## 4. TF-IDF

TF-IDF berfungsi untuk mengetahui bobot suatu kata (*term*) dalam sebuah dokumen. Bobot tersebut dapat digunakan untuk mengetahui apakah kata tersebut bersifat informatif atau tidak. Suatu kata dapat dikatakan informatif, jika nilai TF-IDF mendekati angka 1. Nilai TF-IDF akan diperoleh setelah tahap perhitungan *term frequency* dan *inverse document frequency* dilakukan. Di bawah ini merupakan rumus beserta contoh perhitungan nilai TF-IDF.

$$TF - IDF = TF * IDF \quad (2 . 3)$$

$tf$  = Term Frequency

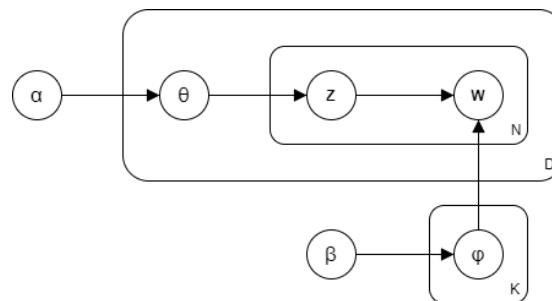
$idf$  = Inverse Document Frequency

**Tabel 2.12** TF-IDF

Term (t)	<b>D1</b>	<b>D2</b>	<b>idf</b>	<b>idf</b>	
				<b>D1</b>	<b>D2</b>
Natural	0	0.167	$\log(2/1) = 0.301$	0	0.301
Recommend	0.5	0.333	$\log(2/2) = 0$	0	0
System	0.5	0.5	$\log(2/2) = 0$	0	0

### 2.1.8 Latent Dirichlet Allocation

*Latent Dirichlet Allocation* (LDA) adalah model probabilistik generatif dari sebuah *corpus*. LDA didasarkan pada asumsi pertukaran sederhana untuk kata-kata dan topik dalam sebuah dokumen. Ide dasar dari model ini adalah bahwa dokumen direpresentasikan sebagai campuran acak terhadap topik laten, dimana setiap topik ditandai oleh distribusi terhadap kata-kata. LDA dapat dikatakan sebagai model pengembangan dari metode LSI dan pLSI dalam hal teknik *dimensionality reduction* [10]. Selain itu juga, LDA merupakan model *unsupervised learning* sehingga setiap topik yang diasosiasikan untuk setiap *class* tidak bisa diberi label. Gambar 2.1 merupakan gambar yang merepresentasikan model untuk LDA.



**Gambar 2.1** Model LDA

Berikut merupakan penjelasan dari bentuk-bentuk serta variabel-variabel yang terdapat pada Gambar 2.1.

1. Lingkaran

Lingkaran merupakan bentuk yang menunjukkan variabel-variabel yang akan digunakan selama proses LDA dilakukan.

2. Tanda Panah

Tanda panah merupakan tanda yang menunjukkan ketergantungan antar 1 variabel dengan variabel yang lainnya.

3. Kotak

Kotak merupakan bentuk yang menunjukkan bahwa bagian tersebut akan dilakukan perulangan (*looping*).

**Tabel 2.13** Deskripsi Model LDA

Variabel	Deskripsi
D	Jumlah dokumen
N	Jumlah semua kata yang ada dalam setiap dokumen
K	Jumlah topik
W	Jumlah setiap kata dalam setiap dokumen
Z	Topik untuk setiap kata dalam setiap dokumen
$\theta$	Distribusi topik untuk dokumen tertentu
$\alpha$	Bobot nilai untuk distribusi dokumen terhadap topik
$\beta$	Bobot nilai untuk distribusi topik terhadap kata
$\phi$	Distribusi kata untuk topik tertentu

Berikut merupakan penjelasan proses dan *pseudocode* dari model *Latent Dirichlet Allocation* [10].

1. Menentukan jumlah topik yang akan digunakan sebagai jumlah label K yang unik dalam *corpus*.
2. Mencari nilai *multinomial topic distributions* variabel  $\phi_k$  untuk setiap topik k, dari *Dirichlet prior*  $\beta$ .
3. Mencari *multinomial mixture distribution*  $\theta^{(d)}$  terhadap semua topik K, untuk setiap dokumen d, dari *Dirichlet prior*  $\alpha$ .
4. Menentukan nilai z dengan menghitung *multinomial distribution* dari parameter  $\theta^{(d)}$  dan tentukan juga nilai w dengan menghitung *multinomial distribution* dari parameter  $\beta_z$  untuk sebanyak kata dalam setiap dokumen.

```

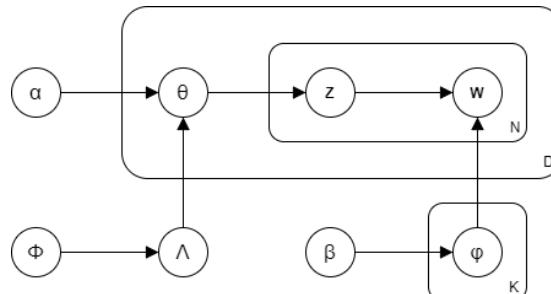
1 For each topic  $k \in \{1, \dots, K\}$ :
2     Generate  $\Phi_k = (\varphi_{k,1}, \dots, \varphi_{k,V})^T \sim \text{Dir}(\cdot | \beta)$ 
3     For each document  $d$ :
4         Generate  $\Theta^{(d)} = (\theta_{l_1}, \dots, \theta_{l_{M_d}})^T \sim \text{Dir}(\cdot | \alpha)$ 
5         For each  $i$  in  $\{1, \dots, N_d\}$ :
6             Generate  $z_i \in \{\lambda_1^{(d)}, \dots, \lambda_{M_d}^{(d)}\} \sim \text{Mult}(\cdot | \Theta^{(d)})$ 
7             Generate  $w_i \in \{1, \dots, V\} \sim \text{Mult}(\cdot | \varphi_{z_i})$ 

```

**Pseudocode 2.1 LDA**

### 2.1.9 Labeled Latent Dirichlet Allocation

*Labeled Latent Dirichlet Allocation* (LLDA) merupakan model grafis probabilistik yang menggambarkan sebuah proses pembuatan dokumen berlabel. LLDA juga merupakan perluasan metode antara LDA (menggabungkan *supervision*) dan *Multinomial Naïve Bayes* (menggabungkan *mixture model*) [11]. Dalam pembelajaran mesin, LLDA merupakan metode yang masuk ke dalam kategori *supervised learning*. LLDA memodelkan sebuah dokumen dalam korpus sebagai campuran topik dan kata-kata yang dihasilkan oleh topik. Lalu, LLDA menyesuaikan antara topik laten dan label (korespondensi satu-satu), dimana label ini dapat dipelajari dan label tersebut merepresentasikan sebuah *class* [4].



**Gambar 2.2 Model LLDA**

Gambar 2.2 merupakan gambar yang merepresentasikan model untuk LLDA. Fungsi dari setiap bentuk yang terdapat pada model LLDA sama seperti fungsi yang terdapat pada model LDA (Gambar 2.1). Berikut merupakan penjelasan variabel-variabel yang terdapat pada model LLDA.

**Tabel 2.14** Deskripsi Model LLDA

Variabel	Deskripsi
$\Lambda$	Sekumpulan biner yang menunjukkan ada / tidaknya topik tersebut dalam sebuah dokumen
$\phi$	Indikator ada / tidaknya topik-topik untuk sebuah dokumen

Berikut merupakan penjelasan proses dan *pseudocode* dari model *Labeled Latent Dirichlet Allocation* (Gambar 2.2) [9].

1. Menentukan jumlah topik yang akan digunakan sebagai jumlah label K yang unik dalam *corpus*.
2. Mencari nilai *multinomial topic distributions* variabel  $\varphi_k$  untuk setiap topik k, dari *Dirichlet prior*  $\beta$ .
3. Dalam model LDA, langkah selanjutnya yang dilakukan adalah mencari *multinomial mixture distribution*  $\theta^{(d)}$  terhadap semua topik K, untuk setiap dokumen d, dari *Dirichlet prior*  $\alpha$ . Dalam model LLDA, terdapat batasan untuk nilai  $\theta^{(d)}$ , dimana nilai tersebut harus didefinisikan hanya terhadap topik yang sesuai dengan labelnya  $\Lambda^{(d)}$ . Jadi langkah selanjutnya yang harus dilakukan adalah mencari nilai  $\Lambda^{(d)}$  dengan melihat label dari setiap dokumen menggunakan *Bernoulli coin toss* untuk setiap topik k, dengan *labeling prior probability*  $\Phi_k$ .
4. Menentukan vektor dari label setiap dokumen menjadi  $\lambda^{(d)} = \{k \mid \Lambda_k^{(d)} = 1\}$ .
5. Nilai  $\lambda^{(d)}$  pada langkah 4 digunakan untuk mendefinisikan proyeksi matriks dari label setiap dokumen secara spesifik dengan persamaan  $L^{(d)} = M_d \times K$  untuk setiap dokumen d, dimana  $M_d = |\lambda^{(d)}|$ . Di bawah ini merupakan *rule* untuk menentukan nilai  $L^{(d)}$  dengan baris  $i \in \{1, \dots, M_d\}$  dan kolom  $j \in \{1, \dots, K\}$ .

$$L_{ij}^{(d)} = \begin{cases} 1, & \text{if } \lambda_i^{(d)} = j. \\ 0, & \text{if } \lambda_i^{(d)} \neq j. \end{cases}$$

$$\lambda^{(d)} = \text{Vektor Label}$$

Contoh: misalkan  $K = 4$  dan dokumen d memiliki label yang diberikan oleh  $\Lambda^{(d)} = \{1, 0, 0, 1\}$ . Nilai  $\Lambda^{(d)}$  menghasilkan nilai  $\lambda^{(d)} = \{1, 4\}$ . Setelah mendapat nilai dari  $\lambda^{(d)}$ , maka nilai  $L^{(d)}$  akan menjadi :

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

6. Gunakan nilai dari matriks  $L^{(d)}$  untuk memproyeksikan vektor dengan parameter *Dirichlet topic prior*  $\alpha$ , dimana hasil proyeksi ini akan menghasilkan nilai  $\alpha^{(d)}$  (vektor dengan dimensi yang lebih kecil) dengan banyak nilai  $M_d = \sum_{k=1}^K \Lambda_k^{(d)}$ .

$$\alpha^{(d)} = L^{(d)} x \alpha = (\alpha_{\lambda_i^{(d)}}, \dots, \alpha_{\lambda_{M_d}^{(d)}})^T$$

$L^{(d)}$  = Proyeksi Matriks Label

7. Menentukan nilai  $\theta^{(d)}$  dengan menghitung *Dirichlet distribution* dari parameter  $\alpha^{(d)}$ .
8. Menentukan nilai z dengan menghitung *multinomial distribution* dari parameter  $\theta^{(d)}$  dan tentukan juga nilai w dengan menghitung *multinomial distribution* dari parameter  $\beta_z$  untuk sebanyak kata dalam setiap dokumen.

```

1 For each topic k ∈ {1, ..., K}:
2     Generate φ_k = (φ_{k,1}, ..., φ_{k,V})^T ~ Dir(. | β)
3     For each document d:
4         For each topic k ∈ {1, ..., K}:
5             Generate Λ^{(d)} k ∈ {0,1} ~ Bernoulli(. | φ_k)
6             Generate α^{(d)} = L^{(d)} x α
7             Generate θ^{(d)} = (θ_{l_1}, ..., θ_{l_{M_d}})^T ~ Dir(. | α^{(d)})
8             For each i in {1, ..., N_d}:
9                 Generate z_i ∈ {λ_1^{(d)}, ..., λ_{M_d}^{(d)}} ~ Mult(. | θ^{(d)})
10                Generate w_i ∈ {1, ..., V} ~ Mult(. | φ_{z_i})

```

#### **Pseudocode 2.2 LLDA**

##### **2.1.10 Simplified Labeled Latent Dirichlet Allocation Classifier (SLLDA-C)**

*Simplified Labeled Latent Dirichlet Allocation Classifier* (SLLDA-C) merupakan metode *Labeled Latent Dirichlet Allocation* yang prosesnya dibuat menjadi lebih sederhana. Di bawah ini merupakan langkah proses dari SLLDA-C [4].

1. Dapatkan nilai  $\Lambda$  dari data *training*. Nilai  $\Lambda$  bisa didapat langsung dari data *training*, karena setiap dokumen dalam *corpus* pasti memiliki 1 label.

2. Lakukan pembelajaran (*learning*) terhadap data *training* untuk mendapatkan nilai  $\phi$  menggunakan *Collapsed Gibbs Sampling* dengan menggunakan nilai  $\alpha$  dan  $\beta$  yang telah ditentukan serta nilai  $\Lambda$  yang telah dipelajari terlebih dahulu pada langkah 1.
3. Setelah nilai  $\phi$  dipelajari dari data *training*, ekstrak 20% kata dengan probabilitas tertinggi untuk setiap topik dari  $\phi$ .
4. Tetapkan label  $k$  ke dokumen jika dokumen tersebut memiliki kata-kata yang paling berkaitan dengan topik  $k$ .

### 2.1.11 *Gibbs Sampling*

*Gibbs Sampling* merupakan algoritma *Markov Chain Monte Carlo* (MCMC) untuk memperoleh serangkaian pengamatan berdasarkan distribusi *multivariate* tertentu. Algoritma ini digunakan ketika pengambilan *sample* secara langsung sulit untuk dilakukan. Dalam LDA, *Gibbs Sampling* digunakan untuk menyederhanakan persamaan matematika yang berfungsi untuk melakukan pembelajaran (*learning*) serta pengambilan keputusan (*inference*) [12]. Di bawah ini merupakan rumus dari *Gibbs Sampling*.

$$p(z_i = j \mid z_{-i}^{(d)}, w^{(d)}) \propto p(z_i = j, w_i = t \mid z_{-i}^{(d)}, w_{-i}^{(d)}) \\ = E(\theta_{dj}) \cdot E(\phi_{jt}) \quad (2.4)$$

$$E(\phi_{jt}) = \frac{n_{-i,j}^{(t)} + \beta_t}{W} \quad (2.5)$$

$\beta_t$  = Nilai Beta untuk Kata ke-t

$n_{-i,j}^{(t)}$  = Jumlah Kata t pada Topik j Dikurangi 1

$$E(\theta_{dj}) = \frac{n_{-i,j}^{(d)} + \alpha_j^{(d)}}{T} \quad (2.6)$$

$\alpha_j^{(d)}$  = Nilai Alpha untuk Topik ke-j Pada Dokumen d

$n_{-i,j}^{(d)}$  = Jumlah Kata di Dokumen d Pada Topik j Dikurangi 1

$$W = \sum_{t=1}^V n_{-i,j}^{(t)} + \sum_{l=1}^V \beta_l \quad (2 . 7)$$

$\beta_l$  = Nilai Beta untuk Kata ke-l

$n_{-i,j}^{(t)}$  = Jumlah Kata t pada Topik j Dikurangi 1

$$T = \sum_{k \in \lambda^{(d)}} n_{-i,k}^{(d)} + \sum_{k=1}^{M_d} \alpha_k^{(d)} \quad (2 . 8)$$

$\alpha_j^{(d)}$  = Nilai Alpha untuk Topik ke-j Pada Dokumen d

$n_{-i,j}^{(d)}$  = Jumlah Kata di Dokumen d Pada Topik j Dikurangi 1

Berikut merupakan penjelasan proses dan *pseudocode* dari algoritma *Gibbs Sampling* [13].

1. Pilih kata beserta topik untuk dilakukan pembelajaran.
2. Berasumsi bahwa kata pada topik tersebut belum pernah masuk sebelumnya. Untuk melakukan hal tersebut maka lakukan pengurangan (*decrement*) terhadap jumlah dokumen terhadap topik dan topik terhadap kata.
3. Hitung nilai probabilitas dari kata tersebut terhadap jumlah topik yang ada pada setiap dokumen.
4. Tetapkan kata tersebut dengan topik yang memiliki probabilitas tertinggi.
5. Lakukan penambahan (*increment*) untuk jumlah dokumen terhadap topik dan jumlah topik terhadap kata.
6. Lakukan langkah 1-5 hingga semua kata telah dilakukan pembelajaran.
7. Lakukan langkah 1-6 hingga iterasi *Gibbs Sampling* selesai.

```

1 For each iteration:
2     For each document d:
3         For each i in {0, ..., N-1}:
4             word <- w[i]
5             topic <- z[i]
6             nd,topic -= 1
7             nword,topic -= 1
8             For each k in {0, ..., Md-1}:
9                 p(z = k.) = E(θdj).E(φjt)
10                topic <- sample dari p(z.)
11                nd,topic += 1
12                nword,topic += 1

```

*Pseudocode 2.3 Gibbs Sampling*

### 2.1.12 F-Measure

Tingkat akurasi merupakan salah satu aspek yang penting dalam hal kategorisasi. Suatu sistem kategorisasi dapat dikatakan baik, jika hasil kategori yang dihasilkan oleh sistem memiliki tingkat akurasi yang tinggi. Untuk mengukur tingkat akurasi dari sistem kategorisasi, terdapat beberapa metode perhitungan untuk menghitung seberapa akurat hasil kategori yang dihasilkan oleh sistem. Metode perhitungan tersebut di antaranya menggunakan *precision* dan *recall* [4]. Di bawah ini merupakan rumus-rumus untuk mengukur nilai akurasi.

#### 1. Precision

$$P_i = \frac{TP_i}{TP_i + FP_i} \quad (2 . 9)$$

$P_i$  = Precision

$TP_i$  = True Positive

$FP_i$  = False Positive

#### 2. Recall

$$R_i = \frac{TP_i}{TP_i + FN_i} \quad (2 . 10)$$

$R_i$  = Recall

$TP_i$  = True Positive

$FN_i$  = False Negative

3. *F-Measure*

$$F - Measure = \frac{2 * (P * R)}{P + R} \quad (2 . 11)$$

*P* = Precision

*R* = Recall

Berikut merupakan penjelasan mengenai *true positive*, *false negative*, dan *false positive*.

1. *True Positive* merupakan nilai pada baris dan kolom yang sama.
2. *False Negative* merupakan penjumlahan nilai pada baris yang sesuai (tanpa mengikutsertakan nilai *true positive*).
3. *False Positive* merupakan penjumlahan nilai pada kolom yang sesuai (tanpa mengikutsertakan nilai *true positive*).

Sebagai contoh, di bawah ini merupakan hasil klasifikasi dari tiga kelas (A, B, dan C) yang akan digunakan untuk mencari nilai *F-Measure*.

**Tabel 2.15** Confusion Matrix

	A	B	C
A	8	2	1
B	3	7	0
C	1	0	5

Untuk melakukan pencarian nilai *F-Measure*, langkah awal yang harus dilakukan adalah mencari nilai *true positive*, *false negative*, dan *false positive* terlebih dahulu. Setelah ketiga nilai tersebut didapatkan, lakukan perhitungan untuk mencari nilai *precision* dan *recall*.

**Tabel 2.16** Contoh Hasil Perhitungan *F-Measure*

	A	B	C
TP	8	7	5
FN	3	3	1
FP	4	2	1
Precision	0.666666667	0.777777778	0.833333333
Recall	0.727272727	0.7	0.833333333

<b>F-Measure</b>	0.695652174	0.736842105	0.833333333
------------------	-------------	-------------	-------------

## 2.2 Tinjauan Studi

Tinjauan studi merupakan bagian yang menjelaskan sumber-sumber pembelajaran yang digunakan selama melakukan peneltian.

### 2.2.1 *State of the Art*

Untuk mendukung penelitian yang dilakukan, dibutuhkan beberapa jurnal yang terkait dengan metode maupun topik penelitian. Jurnal-jurnal ini digunakan sebagai tinjauan penulis dalam melakukan penelitian. Tabel 2.16 merupakan tabel yang merepresentasikan jurnal-jurnal yang digunakan sebagai tinjauan studi.

**Tabel 2.17 State of the Art**

Peneliti	Judul	Tahun	Masalah	Metode
Yiqi Bai, Jie Wang	News Classifications with Labeled LDA	2015	Melakukan klasifikasi terhadap berita dengan melakukan modifikasi terhadap Labeled LDA untuk meningkatkan tingkat akurasi	LLDA-C, SLLDA-C, SVM
Daniel Ramage, David Hall, Ramesh Nallapati, Christopher D. Manning	Labeled LDA: A supervised topic model for credit attribution in multi labeled-corpora	2009	Mencoba menyelesaikan permasalahan multi-tag pada kasus credit attribution dengan menggunakan metode pengembangan dari LDA (LLDA).	LLDA, SVM
Ge Song, Yunming Ye, Xiaolin Du, Xiaohui Huang, Shifu Bie	Short Text Classification: A Survey	2014	Melakukan klasifikasi terhadap teks yang pendek	LSA, pLSA, LDA

## 2.3 Tinjauan Objek

Berikut merupakan objek-objek yang akan digunakan dalam penelitian sistem kategorisasi dokumen.

### 1. Abstraksi

Abstraksi merupakan uraian singkat tetapi lengkap yang berisi judul, permasalahan pendekatan terhadap masalah landsan teoritik yang digunakan, hasil temuan, dan rekomendasi. Fungsi abstrak adalah memberikan gambaran kepada pembaca mengenai isi laporan yang akan dibacanya. Isi abstrak meliputi judul penelitian, rumusan masalah penelitian, metode penelitian, teknik dan pengumpulan data penelitian serta hasil dan kesimpulan peneltian yang telah dibuat

### 2. Jurnal Penelitian

Jurnal penelitian merupakan sebuah laporan peneliti tentang hasil penelitian yang telah dilakukan secara ilmiah atau merupakan kesimpulan dari penelitian skripsi yang telah dilakukan. Pada dasarnya, sebagian besar jurnal penelitian dapat dipertanggungjawabkan keilmiahannya tergantung dari metode yang dipakai dalam pembuatan dan penyusunan laporan jurnal penelitian. Biasanya laporan jurnal penelitian dimasukkan dalam terbitan kumpulan jurnal bersama-sama dengan laporan peneliti lain. Contoh sumber untuk mendapat jurnal penelitian diantaranya dapat dilihat pada website IEEE, ACM, dan *Research Gate*.

## **BAB III**

### **ANALISIS DAN PERANCANGAN**

#### **3.1 Analisis Masalah**

Dalam penelitian ini, sistem kategorisasi akan memiliki data masukan berupa dokumen yang telah diberi label. Dokumen yang akan digunakan adalah dokumen berbentuk penelitian dengan bagian abstrak sebagai fiturnya. Setiap dokumen yang digunakan dapat memiliki lebih dari 1 label (*multi label*). Label yang diberikan pada setiap dokumen adalah label berupa kategori untuk jenis dokumen penelitian. Label yang digunakan untuk dokumen penelitian adalah *machine learning*, *natural language processing*, dan *image processing*. Data masukan dokumen berabel ini akan digunakan oleh sistem untuk melakukan pengolahan data agar sistem dapat berpikir dan mengambil keputusan seperti layaknya manusia. Jumlah data yang akan digunakan adalah 200 data. Dokumen penelitian akan dibagi menjadi 2 bagian yaitu data *training* dan data *testing*. Jumlah data yang digunakan untuk data *training* adalah 180 data, sedangkan jumlah data untuk data *testing* adalah 20 data.

Untuk dapat melakukan hal tersebut, setiap dokumen akan melalui tahap *pre-processing* terlebih dahulu. Tahap *pre-processing* ini merupakan tahap untuk memecah sebuah kalimat menjadi sekumpulan kata-kata yang memiliki makna tertentu. *Pre-processing* akan dimulai dari melakukan *case folding*, *filtering*, *tokenizing*, *stopword removing*, sampai *stemming*.

Setelah menjadi sekumpulan kata-kata, langkah selanjutnya adalah melakukan pemilihan kata yang telah di-*pre-processing* untuk mengurangi kata-kata yang bobot nilainya rendah. Pemilihan kata-kata ini dapat dilakukan dengan cara pembobotan. Pembobotan ini bertujuan untuk mengurangi jumlah kata yang akan digunakan sistem ketika melakukan pengolahan data agar waktu yang dibutuhkan sistem dapat berkurang. Teknik pembobotan yang akan digunakan dalam penelitian ini adalah TF-IDF.

Setelah melalui tahap pembobotan, kata-kata yang telah terpilih akan diolah kembali untuk dilakukan pembelajaran (*learning*) menggunakan salah satu algoritma *machine learning* yaitu *Labeled Latent Dirichlet Allocation*. Pembelajaran ini bertujuan untuk menghasilkan suatu model, dimana model ini kemudian akan dijadikan suatu dasar dalam melakukan pengambilan keputusan. Dalam proses pembelajaran menggunakan *Labeled Latent Dirichlet Allocation* terdapat algoritma tambahan untuk mempermudah rumus perhitungan probabilitas suatu kata. Algoritma tambahan yang digunakan dalam penelitian ini adalah *Gibbs Sampling*. Model yang dihasilkan oleh algoritma *Labeled Latent Dirichlet Allocation* adalah berupa matriks probabilitas dokumen terhadap topik ( $\Theta$ ) dan matriks probabilitas topik terhadap kata ( $\varphi$ ).

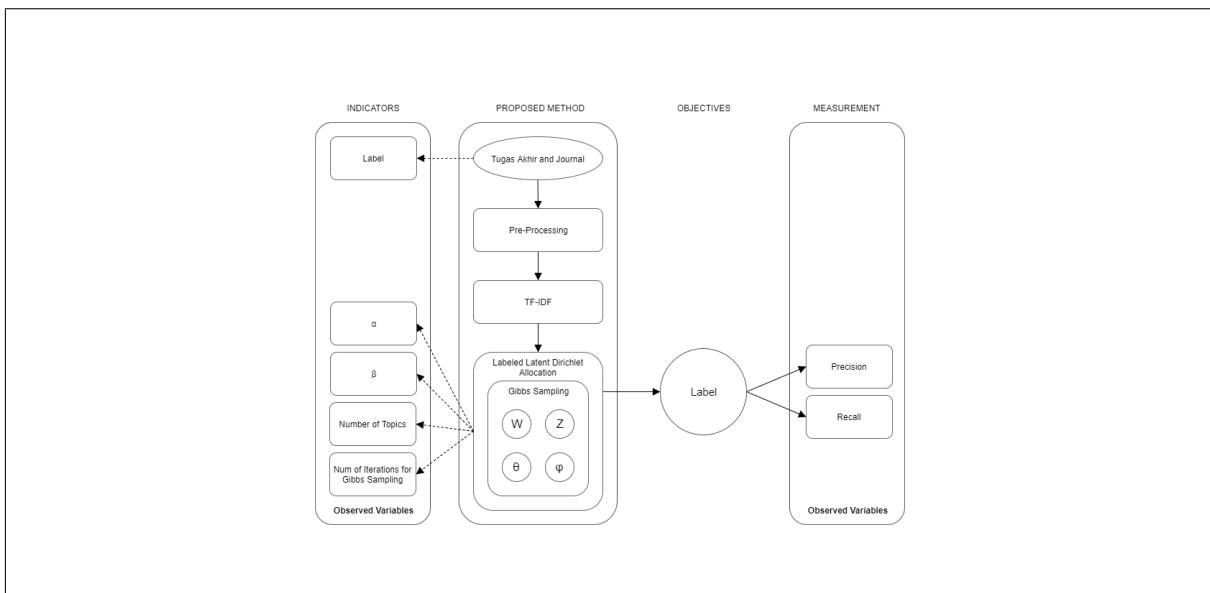
Setelah proses pembelajaran selesai, model yang telah dihasilkan dapat digunakan

untuk proses pengambilan keputusan. Dalam penelitian ini, langkah algoritma yang digunakan untuk pengambilan keputusan adalah *Simplified Labeled Latent Dirichlet Allocation*. Algoritma ini melakukan pengambilan keputusan hanya dengan menggunakan matriks probabilitas topik terhadap kata saja. Sesuai dengan namanya (*simplified*), algoritma ini mengambil keputusan tersebut dengan cara melihat kemunculan jumlah kata terbanyak dari dokumen yang diuji dengan kata-kata yang ada di matriks probabilitas topik terhadap kata (20% kata dengan probabilitas tertinggi pada setiap topik). Jumlah kemunculan kata terbanyak ini bertujuan untuk mengetahui keterkaitan antara kata-kata pada dokumen yang diuji dengan hasil kata-kata pada setiap topik yang telah dilakukan pembelajaran. Setelah mendapatkan kata-kata yang paling berkaitan dengan topik tertentu, maka tetapkan topik tersebut pada dokumen yang sedang diuji. Hasil label dari sistem dapat memiliki lebih dari 1 label (*multi-label*) juga.

#### 3.2 Kerangka Pemikiran

Gambar 3.1 merupakan gambar yang merepresentasikan kerangka pemikiran dari sistem kategorisasi dokumen yang akan diteliti. Di bawah ini merupakan penjelasan dari kerangka pemikiran pada Gambar 3.1.

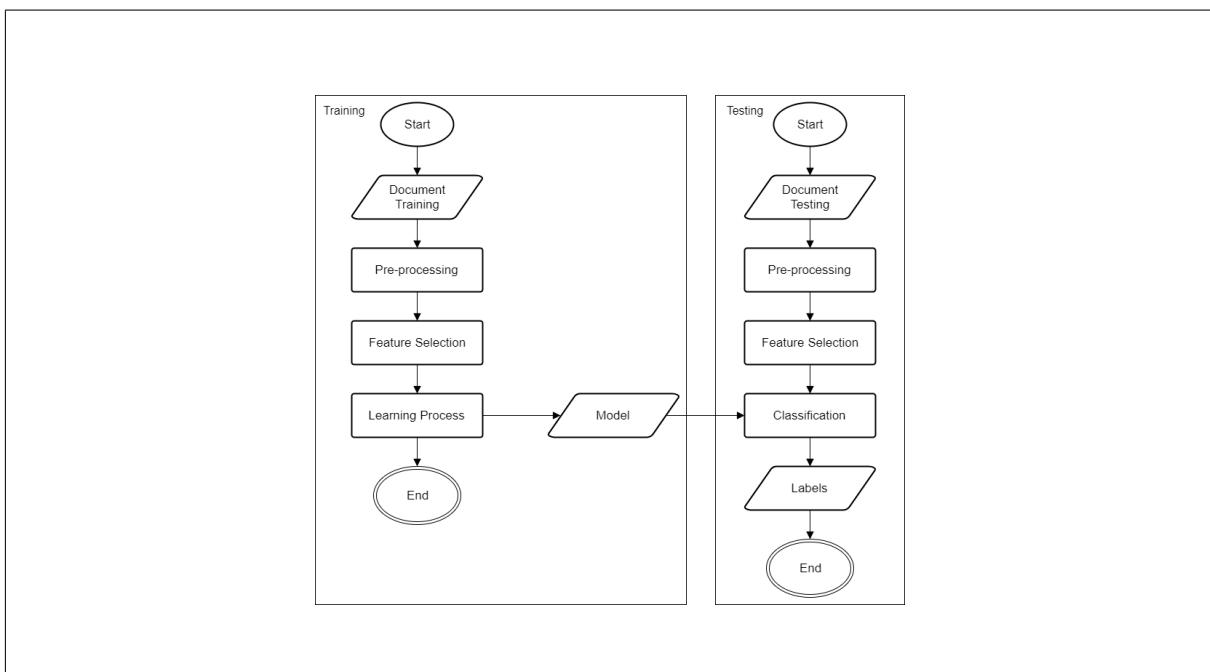
1. Proses kategorisasi akan dimulai dengan masukan dataset yang berupa dokumen penelitian, dimana setiap data telah diberikan topik/label terlebih dahulu.
2. Setelah itu, dokumen penelitian yang telah memiliki label akan dilakukan proses *pre-processing*.
3. Ketika semua data telah melewati proses *pre-processing*, maka kata-kata yang terdapat pada semua dokumen akan dihitung bobot nilainya dengan TF-IDF. Perhitungan nilai bobot ini berfungsi untuk menyeleksi kata-kata apa saja yang akan digunakan ke dalam proses klasifikasi.
4. Kata-kata yang terpilih serta label yang telah ditetapkan pada tahap awal kemudian akan diolah untuk dilakukan klasifikasi dengan metode *Labeled Latent Dirichlet Allocation*.
5. Pada bagian metode *Labeled Latent Dirichlet Allocation* terdapat algoritma *Gibbs Sampling* yang berfungsi untuk melakukan learning dan *inference* terhadap data yang telah diolah oleh metode *Labeled Latent Dirichlet Allocation*. Untuk dapat melakukan *learning* dan *inference* terdapat beberapa indikator yang perlu ditentukan terlebih dahulu. Indikator ini berupa nilai  $\alpha$ ,  $\beta$ , jumlah topik, dan jumlah iterasi untuk *Gibbs Sampling*. Selain itu juga, terdapat beberapa nilai variabel seperti  $W$ ,  $Z$ ,  $\theta$ , dan  $\phi$  yang harus dihitung untuk dapat melakukan *learning* dan *inference* menggunakan *Gibbs Sampling*.
6. Hasil dari pengolahan Labeled LDA dan *Gibbs Sampling* adalah berupa topik/kategori.
7. Hasil pada bagian 6 dapat diukur dengan menghitung nilai akurasi yang dihasilkan terhadap dokumen penelitian yang diuji. Pengukuran akurasi ini dapat dilakukan dengan menghitung nilai *precision* dan *recall*.



Gambar 3.1 Kerangka Pemikiran LLDA

### 3.3 Perancangan

Perancangan merupakan bagian yang menjelaskan urutan proses suatu sistem mulai dari masukan sampai ke keluaran. Pada bagian ini, setiap proses yang telah dijelaskan akan dianalisis secara manual agar lebih mudah untuk memahami setiap proses yang berlangsung selama sistem berjalan. Di bawah ini merupakan *flowchart* (Gambar 3.2) beserta penjelasan setiap proses untuk sistem klasifikasi dokumen berdasarkan analisis masalah.



Gambar 3.2 Flowchart Global

### 3.3.1 Data Sampling

Data yang akan digunakan untuk penelitian merupakan data yang bersumber dari IEEE, ACM, *Research Gate*, dan kampus Institut Teknologi Harapan Bangsa. Data yang didapat kemudian akan direpresentasikan ke dalam *file .xls* agar lebih mudah dalam pembacaan (terstruktur) dan perubahan data. Di bawah ini merupakan contoh data *training* maupun *testing* yang telah direpresentasikan ke dalam *file .xls*.

**Tabel 3.1** Contoh Data dalam *File .xls*

Title	Label	Content	Author
LDA	ML, NLP	Automatically categorizing news articles with high accuracy is an important task in an automated quick news system.	Blei
LLDA	ML, NLP	This paper introduces Labeled LDA, a topic model that constrains Latent Dirichlet Allocation by defining a one-to-one correspondence between LDA's latent topics and user tags.	Ramage

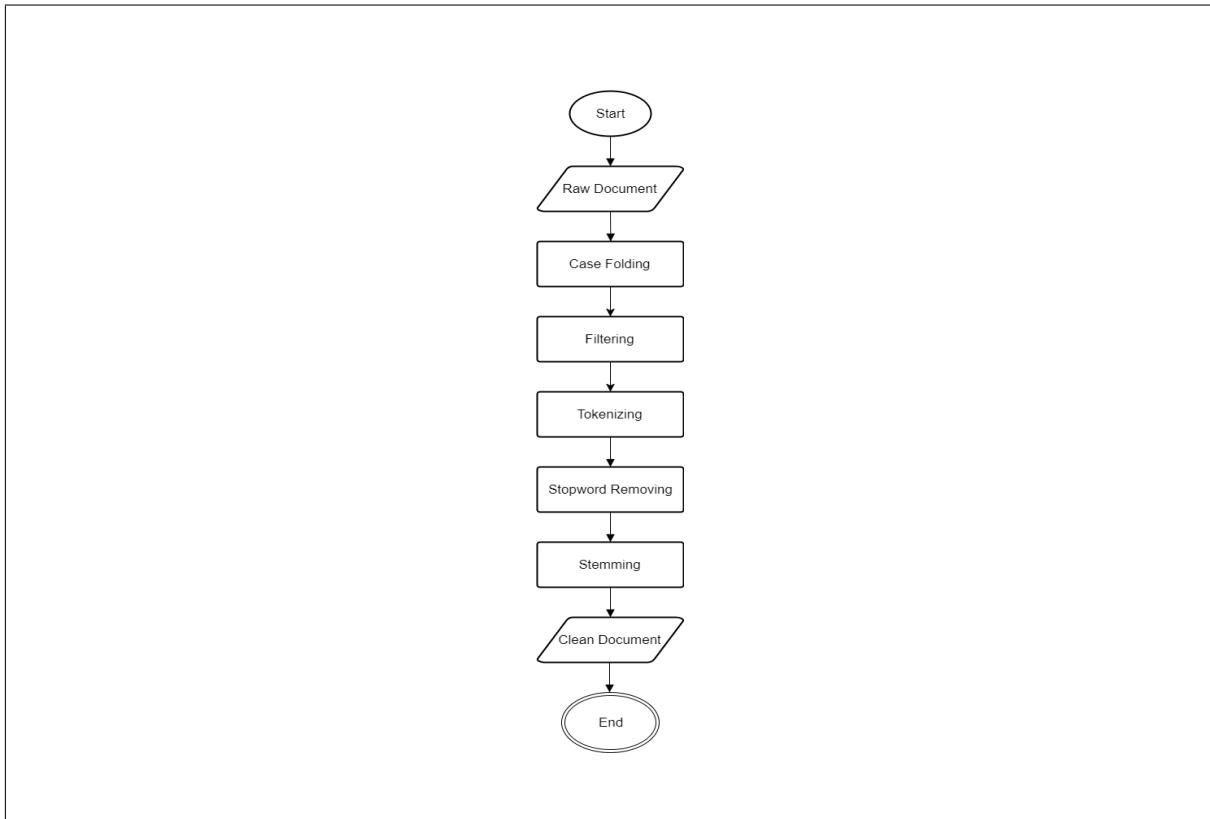
Berikut merupakan karakteristik data abstrak :

1. Menjelaskan masukan, proses, dan keluaran suatu sistem.
2. Sering muncul kata-kata “menyarankan” atau “menawarkan” (*proposed*) suatu metode.
3. Terdapat kalimat mengenai saran pengembangan di kemudian hari.

**Tabel 3.2** Jumlah Data *Training* dan *Testing*

	<b>Data Training</b>	<b>Data Testing</b>
NLP	40	5
Machine Learning	25	3
Image Processing	54	5
NLP dan Machine Learning	43	5
Machine Learning dan Image Processing	18	2

### 3.3.2 *Pre-Processing*



**Gambar 3.3 Flowchart Pre-Processing**

Pada tahap ini dokumen yang digunakan akan diolah menjadi sekumpulan kata-kata (*token*) yang kemudian akan digunakan untuk pengolahan data. Keluaran dari tahap *pre-processing* merupakan daftar kata yang telah diolah menjadi bentuk dasar dan sudah tidak terdapat tanda baca, kata penghubung, kata depan, dll. Untuk melakukan hal tersebut, langkah yang akan dilakukan dalam penelitian ini adalah melakukan *case folding*, *filtering*, *tokenizing*, *stopword removing*, dan *stemming*. Gambar 3.3 merupakan alur proses dari tahap *pre-processing*.

#### 3.3.2.1 *Case Folding*

*Case Folding* digunakan untuk mengurangi jumlah fitur kata yang digunakan. Berdasarkan contoh data pada Tabel 3.1, bagian content dalam data akan dilakukan *case folding*. Tabel 3.3 merupakan data yang telah melalui tahap *case folding*.

**Tabel 3.3 Hasil Case Folding**

Title	Label	Content	Author
LDA	ML, NLP	automatically categorizing news articles with high accuracy is an important task in an automated quick news system.	Blei
LLDA	ML, NLP	this paper introduces labeled lda, a topic model that constrains latent dirichlet allocation by defining a one-to-one correspondence between lda's latent topics and user tags.	Ramage

### 3.3.2.2 Filtering

Pada tahap *filtering*, penghapusan yang dilakukan adalah penghapusan terhadap tanda baca, *tag*, *email*, spasi lebih dari 1, dan *link*. *Filtering* digunakan untuk mengurangi jumlah fitur kata yang digunakan. *Filtering* dapat dilakukan dengan menggunakan *regular expression*. Sebagai contoh berdasarkan Tabel 3.3, bagian *content* dari setiap dokumen akan dilakukan *filtering*. Tabel 3.4 merupakan contoh data yang telah melalui tahap *filtering*.

**Tabel 3.4 Hasil Filtering**

Title	Label	Content	Author
LDA	ML, NLP	automatically categorizing news articles with high accuracy is an important task in an automated quick news system	Blei
LLDA	ML, NLP	this paper introduces labeled lda a topic model that constrains latent dirichlet allocation by defining a one to one correspondence between lda s latent topics and user tags	Ramage

### 3.3.2.3 Tokenizing

Pada tahap ini, kata-kata yang telah melalui tahap *filtering* akan dipecah menjadi kumpulan kata-kata yang disebut sebagai *token*. Jenis *tokenizing* yang digunakan pada penelitian ini adalah *word tokenize*. Sebagai contoh, jika bagian *content* pada Tabel 3.4 untuk dokumen 1 dilakukan *tokenizing*, maka hasil yang akan dihasilkan adalah seperti pada Tabel 3.5.

**Tabel 3.5** Hasil *Tokenizing*

<b>Kumpulan Kata-kata</b>		
automatically	categorizing	news
articles	with	high
accuracy	is	an
automated	quick	news
system		

### 3.3.2.4 *Stopword Removing*

Pada tahap ini, kata-kata yang telah melalui tahap *tokenizing* akan dilakukan proses penghapusan terhadap kata-kata yang tidak mewakili makna dari suatu kalimat. *Stopword removing* bertujuan untuk mengurangi jumlah fitur yang digunakan. Daftar *stopword* yang digunakan merupakan *stopword* yang digunakan Google dan bersumber dari www.ranks.nl. Sebagai contoh, Tabel 3.6 merupakan hasil *stopword removing* terhadap contoh kata-kata pada Tabel 3.5.

**Tabel 3.6** Hasil *Stopword Removing*

<b>Kumpulan Kata-kata</b>		
automatically	categorizing	news
articles	accuracy	automated
quick	news	system

### 3.3.2.5 *Stemming*

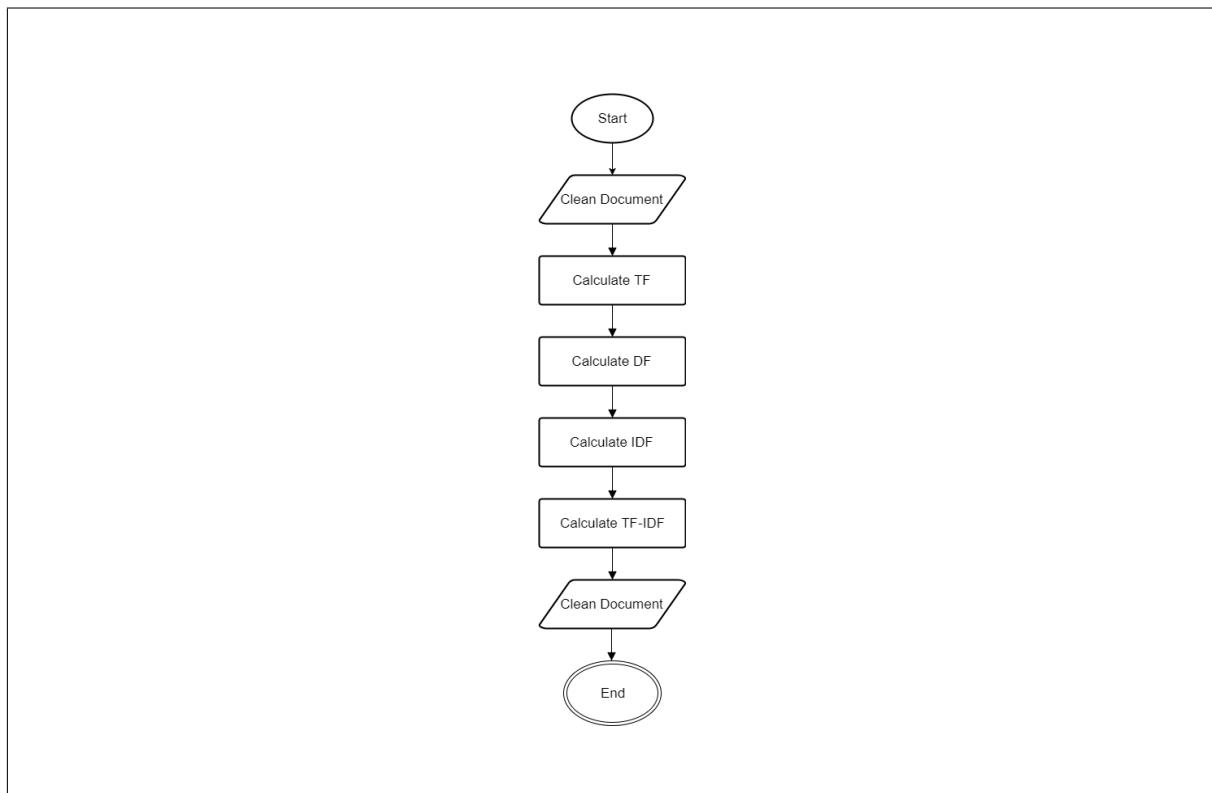
Pada tahap ini, kata-kata yang telah melalui tahap *stopword removing* akan diubah menjadi kata dasar sesuai dengan pembentuk katanya. *Stemming* bertujuan untuk mengurangi jumlah fitur yang digunakan. Proses *stemming* untuk dokumen berbahasa Inggris akan dilakukan dengan menggunakan algoritma *Snowball Stemmer*. Tabel 3.7 merupakan contoh hasil *stemming* terhadap contoh kata-kata pada Tabel 3.6.

**Tabel 3.7 Hasil Stemming**

<b>Kumpulan Kata-kata</b>		
automatic	categorize	news
article	accuracy	automate
quick	news	system

### 3.3.3 Feature Selection

Pada tahap ini, kata-kata yang telah di *pre-processing* akan dihitung bobot nilai dari setiap katanya. Pembobotan ini bertujuan untuk mengurangi jumlah kata yang digunakan selama melakukan pengolahan data. Teknik pembobotan yang akan digunakan dalam penelitian ini adalah TF-IDF. Proses perhitungan TF-IDF yang akan dilakukan di antaranya adalah menghitung nilai TF, DF, dan IDF dengan *unigram* sebagai jenis *bag of words*-nya. Gambar 3.4 merupakan alur proses dari penggunaan TF-IDF.

**Gambar 3.4 Flowchart Feature Selection**

#### 3.3.3.1 Term Frequency

Pada tahap ini, setiap kata dalam suatu dokumen akan dihitung jumlah katanya. Dengan menggunakan Persamaan 2.1, maka kumpulan kata yang telah melalui tahap *pre-processing* akan digunakan untuk mendapatkan nilai TF. Sebagai contoh pada Tabel 3.8 terdapat 2 dokumen yang telah melalui tahap *pre-processing*.

**Tabel 3.8** Contoh *Term Count*

Document 1		Document 2	
Term	Term Count	Term	Term Count
latent	3	latent	2
topic	2	topic	4
news	1	credit	1
article	2	tag	1

Tabel 3.8 merupakan tabel yang menunjukkan jumlah kata tertentu di suatu dokumen. Langkah selanjutnya adalah melakukan normalisasi terhadap jumlah kata tersebut. Normalisasi ini dilakukan agar nilai bobot yang digunakan ketika perhitungan TF-IDF akan berkisar antara 0 sampai 1. Sebagai contoh, jika kata “latent” pada setiap dokumen akan dihitung nilai TF-nya, maka hasilnya adalah sebagai berikut.

$tf("latent", d1) = 3/8 = 0.375$ $tf("latent", d2) = 2/8 = 0.25$
--

Sesuai dengan perhitungan di atas untuk kata “latent”, Tabel 3.9 merupakan nilai TF untuk kata-kata yang lain pada dokumen 1 dan 2.

**Tabel 3.9** Contoh *Term Frequency*

Document 1		Document 2	
Term	Term Count	Term	Term Count
latent	0.375	latent	0.25
topic	0.25	topic	0.5
news	0.125	credit	0.125
article	0.25	tag	0.125

### 3.3.3.2 Document Frequency

*Document Frequency* (DF) merupakan nilai yang menunjukkan jumlah dokumen yang memiliki kata tertentu. Untuk menghitung nilai DF, salah satu cara yang dapat dilakukan adalah menggunakan bilangan biner untuk memberikan suatu informasi terhadap semua kata yang terdapat pada setiap dokumen (1 = ada dan 0 = tidak ada).

### III. ANALISIS DAN PERANCANGAN

---

**Tabel 3.10** Contoh Informasi Kata yang Muncul

	<b>Doc 1</b>	<b>Doc 2</b>
latent	1	1
topic	1	1
news	1	0
article	1	0
credit	0	1
tag	0	1

Setelah memberikan informasi dengan bilangan biner untuk setiap kata pada semua dokumen, langkah selanjutnya adalah menjumlahkan semua bilangan biner pada semua dokumen untuk setiap kata.

**Tabel 3.11** Contoh *Document Frequency*

<b>Term</b>	<b>Doc 1</b>	<b>Doc 2</b>	<b>DF</b>
latent	1	1	2
topic	1	1	2
news	1	0	1
article	1	0	1
credit	0	1	1
tag	0	1	1

#### 3.3.3.3 *Inverse Document Frequency*

*Inverse Document Frequency* (IDF) merupakan nilai yang digunakan untuk meningkatkan bobot nilai suatu kata yang tidak terlalu sering muncul dan mengurangi bobot yang sering muncul. Dengan menggunakan Persamaan 2.2, maka nilai IDF pada contoh tabel 3.12 adalah sebagai berikut.

**Tabel 3.12** Contoh *Inverse Document Frequency*

Term	DF	IDF
latent	2	$\log(2/2) = 0$
topic	2	$\log(2/2) = 0$
news	1	$\log(2/1) = 0.301$
article	1	$\log(2/1) = 0.301$
credit	1	$\log(2/1) = 0.301$
tag	1	$\log(2/1) = 0.301$

### 3.3.3.4 TF-IDF

TF-IDF merupakan nilai yang menunjukkan seberapa informatif suatu kata muncul di setiap dokumen. Semakin tinggi nilai TF-IDF pada suatu kata, maka semakin informatif juga kata tersebut pada suatu dokumen. Jika kata “*latent*” pada setiap dokumen akan dihitung nilai TF-IDF-nya, maka hasilnya adalah sebagai berikut.

tfidf("latent", d1) = 0.375 * 0 = 0
tfidf("latent", d2) = 0.25 * 0 = 0

Sesuai dengan perhitungan di atas untuk kata “*latent*”, Tabel 3.13 merupakan nilai TF-IDF untuk kata-kata yang lain pada dokumen 1 dan 2 pada Tabel 3.8.

**Tabel 3.13** Contoh TF-IDF

Term	TF		IDF	TF-IDF	
	Doc 1	Doc 2		Doc 1	Doc 2
latent	0.375	0.25	0	0	0
topic	0.25	0.5	0	0	0
news	0.125	0	0.301	0.037625	0
article	0.25	0	0.301	0.07525	0
credit	0	0.125	0.301	0	0.037625
tag	0	0.125	0.301	0	0.037625

Setelah mendapatkan nilai TF-IDF, langkah selanjutnya adalah melakukan pemilihan kata berdasarkan nilai TF-IDF tersebut. Pemilihan kata ini dilakukan dengan cara mengambil

n TF-IDF tertinggi. Sebagai contoh, kata akan dipilih dengan mengambil 2 TF-IDF tertinggi. Berdasarkan nilai TF-IDF pada Tabel 3.13, 2 TF-IDF tertinggi terletak pada kata “news”, “article”, “credit”, dan “tag”. Nilai TF-IDF untuk kata “news”, “credit”, dan “tag” adalah bernilai sama, sehingga kata yang akan dipilih adalah kata yang terlebih dahulu dilakukan pengecekan (dilakukan dari dokumen 1 dan untuk kata-kata dicek dari yang paling atas ke bawah). Oleh karena itu, kata yang akan dipilih berdasarkan 2 TF-IDF tertinggi adalah kata “news” dan “article”. Setelah mendapatkan kata-kata yang terpilih, langkah selanjutnya adalah melakukan pemeriksaan setiap kata pada semua dokumen dengan kondisi jika terdapat suatu kata yang tidak terpilih dari 2 TF-IDF tertinggi, maka kata tersebut akan dihapus (selain kata “news” dan “article”).

### 3.3.4 Learning Process

*Learning Process* merupakan bagian yang menjelaskan langkah serta perhitungan dari penggunaan algoritma untuk menghasilkan *classifier model*. Algoritma yang akan digunakan pada penelitian ini adalah *Simplified Labeled Latent Dirichlet Allocation Classifier* (SLLDA-C). Langkah-langkah untuk menggunakan SLLDA-C dapat dilihat pada bab 2 bagian 2.1.10. Berdasarkan langkah SLLDA-C pada bagian 3.3.4 ini, langkah yang akan dijelaskan hanya pada sampai langkah ke-2 saja, karena langkah ke-1 sampai langkah ke-2 merupakan langkah untuk menghasilkan *classifier model* sedangkan langkah ke-3 dan ke-4 merupakan langkah untuk pengujian. Berikut merupakan contoh penggunaan SLLDA-C.

1. Siapkan dokumen yang akan dijadikan sebagai data *training*, dimana setiap dokumen telah diberi label terlebih dahulu. Lalu simpan label beserta isi dokumen ke dalam sebuah *file* (.xls).

**Tabel 3.14** Contoh Isi Dokumen Beserta Label

	Kata-kata	Label
Doc 1	classify, recommend, classify	NLP, ML
Doc 2	image, classify	ML, Citra
Doc 3	text, text	NLP

2. Tetapkan nilai  $\alpha$ ,  $\beta$ , jumlah label (K), dan jumlah iterasi untuk digunakan ketika proses pembelajaran berlangsung. Nilai  $\alpha$  dan  $\beta$  memiliki nilai *default* yaitu  $50/K$  dan  $0.1$  [10].

### III. ANALISIS DAN PERANCANGAN

---

**Tabel 3.15** Contoh Nilai Masukan LLDA

	<b>Default</b>	<b>Nilai yang akan digunakan</b>
K	-	$3 \rightarrow [\text{NLP, ML, Citra}]$
$\alpha$	$50/K$	16.67
$\beta$	0.1	0.1
Jumlah Iterasi	1000	1000

3. Mencari nilai  $\Lambda$  dengan melihat label pada setiap dokumen. Langkah ini merupakan langkah pada *Pseudocode* 2.2 baris 5. Nilai  $\Lambda$  merupakan nilai yang menunjukkan ada tidaknya topik tersebut pada setiap dokumen. Bilangan yang digunakan merupakan bilangan biner, dimana angka 1 menunjukkan bahwa suatu dokumen memiliki topik tersebut dan angka 0 menunjukkan suatu dokumen tidak memiliki topik tersebut.

**Tabel 3.16** Contoh Pemetaan Ada Tidaknya Suatu Topik Pada Setiap Dokumen

$\Lambda^{(d)}$	<b>Nilai</b> $\rightarrow \{ \text{NLP, ML, Citra} \}$
$\Lambda^{(1)}$	{ 1, 1, 0 }
$\Lambda^{(2)}$	{ 0, 1, 1 }
$\Lambda^{(3)}$	{ 1, 0, 0 }

4. Lakukan pengindeksan untuk setiap kata yang unik yang terdapat pada setiap dokumen. Kata yang sama memiliki indeks yang sama.

**Tabel 3.17** Contoh Hasil Pengindeksan Terhadap Semua Kata

	<b>Kata-kata</b>	<b>Indeks Kata</b>
<b>Doc 1</b>	classify	1
	recommend	2
	classify	1
<b>Doc 2</b>	image	3
	classify	1
<b>Doc 3</b>	text	4
	text	4

### III. ANALISIS DAN PERANCANGAN

---

5. Siapkan matriks untuk menampung jumlah topik terhadap kata (*topic-word*), dokumen terhadap topik (*document-topic*), dan *topic assignment* (*document-word*).

**Tabel 3.18** Contoh Matriks Awal Jumlah Topik Terhadap Kata

Topic-Word				
	classify	recommend	image	text
<b>NLP</b>	0	0	0	0
<b>ML</b>	0	0	0	0
<b>Citra</b>	0	0	0	0

**Tabel 3.19** Contoh Matriks Awal Jumlah Dokumen Terhadap Topik

Document-Topic			
	ML	NLP	Citra
<b>Doc 1</b>	0	0	0
<b>Doc 2</b>	0	0	0
<b>Doc 3</b>	0	0	1

**Tabel 3.20** Contoh Awal Penetapan Topik Terhadap Semua Kata

Topic Assignment		
	Kata-kata	Indeks Topik
<b>Doc 1</b>	classify	0
	recommend	0
	classify	0
<b>Doc 2</b>	image	0
	classify	0
<b>Doc 3</b>	text	0
	text	0

6. Menetapkan topik secara acak untuk setiap kata yang terdapat pada setiap dokumen. Ketika setiap kata diberikan sebuah topik, tambahkan (*increment*) jumlah kata dengan topik tersebut pada kolom dan baris yang sesuai pada matriks *topic-word*. Selain itu juga, lakukan

### III. ANALISIS DAN PERANCANGAN

---

perubahan nilai pada matriks *topic assignment* dengan melihat dokumen serta kata yang sesuai dengan indeks topik yang didapat. Langkah ini dilakukan untuk mengganti nilai  $\theta$  yang belum pernah dihasilkan sebelumnya (*Pseudocode 2.2* pada baris 7).

**Tabel 3.21** Contoh Hasil Random Topic Untuk Semua Kata

Topic Assignment		
	Kata-kata	Indeks Topik
<b>Doc 1</b>	classify	2
	recommend	2
	classify	1
<b>Doc 2</b>	image	3
	classify	2
<b>Doc 3</b>	text	2
	text	1

**Tabel 3.22** Contoh Hasil Penambahan Nilai Matriks *Topic-Word*

Topic-Word				
	classify	recommend	image	text
<b>NLP</b>	1	0	0	1
<b>ML</b>	2	1	0	1
<b>Citra</b>	0	0	1	0

- Setelah matriks *topic assignment* dan *topic-word* terisi, langkah selanjutnya adalah mengisi matriks *document-topic*. Nilai dari matriks *document-topic* merupakan jumlah kata pada suatu topik yang terdapat pada setiap dokumen. Untuk mempermudah perhitungan nilai pada matriks *document-topic* gunakan matriks *topic assignment* untuk melihat jumlah katanya. Berdasarkan matriks *topic assignment*, maka hasil dari matriks *document-topic* dapat dilihat pada Tabel 3.23.

### III. ANALISIS DAN PERANCANGAN

---

**Tabel 3.23** Contoh Hasil Penambahan Nilai Matriks *Document-Topic*

Document-Topic			
	ML	NLP	Citra
<b>Doc 1</b>	1	2	0
<b>Doc 2</b>	0	1	1
<b>Doc 3</b>	1	1	0

8. Lakukan pembelajaran (*learning*) terhadap matriks di atas untuk mendapatkan nilai  $\phi$  menggunakan *Collapsed Gibbs Sampling* dengan menggunakan nilai  $\alpha$ ,  $\beta$ , dan  $\Lambda$  yang telah ditentukan. Langkah ini merupakan langkah untuk memperbarui (*update*) topik pada setiap kata yang dilakukan secara acak menjadi topik yang lebih sesuai. Sesuai dengan langkah *Gibbs Sampling* pada *Pseudocode 2.3*, langkah awal yang harus dilakukan adalah memilih kata yang akan diperbarui topiknya. Lalu berasumsi bahwa kata pada topik tersebut belum pernah dihitung sebelumnya. Dalam *Pseudocode 2.2*, langkah ini merupakan *Pseudocode* pada baris 8 dan 9, sedangkan dalam *Pseudocode 2.3* langkah ini merupakan langkah 4 sampai 7.

Contoh:

Kata "classify" ke-2 pada dokumen 1 ingin diperbarui.

**Tabel 3.24** Contoh Pengamsumsian Topik Untuk Kata ke-1 Dokumen 1

Topic Assignment		
	Kata-kata	Indeks Topik
<b>Doc 1</b>	classify	<b>0</b>
	recommend	2
	classify	1
<b>Doc 2</b>	image	3
	classify	2
<b>Doc 3</b>	text	2
	text	1

### III. ANALISIS DAN PERANCANGAN

---

**Tabel 3.25** Contoh Pengurangan Nilai Matriks *Topic-Word* Untuk Kata ke-1 Dokumen 1

Topic-Word				
	classify	recommend	image	text
<b>NLP</b>	1	0	0	1
<b>ML</b>	<b>1</b>	1	0	1
<b>Citra</b>	0	0	1	0

**Tabel 3.26** Contoh Pengurangan Nilai Matriks *Document-Topic* Untuk Topik ke-2 Dokumen 1

Document-Topic			
	ML	NLP	Citra
<b>Doc 1</b>	1	<b>1</b>	0
<b>Doc 2</b>	0	1	1
<b>Doc 3</b>	1	1	0

9. Hitung probabilitas *topic-word* dan *document-topic* dengan menggunakan Persamaan 2.4 atau dalam *Pseudocode* 2.3 langkah ini merupakan langkah pada baris 9. Lakukan langkah ini untuk setiap topik yang ada.

Contoh:

(a) *document-topic* (Left)

left = Jumlah topik k pada dokumen yang diperbarui \*  $\alpha$  / (Jumlah kata pada dokumen yang diperbarui - 1 + (K \*  $\alpha$ ))

(b) *topic-word* (Right)

right = Jumlah kata yang diperbarui pada topik k \*  $\beta$  / (Jumlah semua kata pada topik k + (V \*  $\beta$ ))

Sesuai dengan nilai-nilai yang didapat dan ditetapkan pada langkah sebelumnya, maka nilai probabilitas untuk *document-topic* dan *topic-word* dapat dilihat pada Tabel 3.27.

**Tabel 3.27** Contoh Perhitungan Probabilitas *Topic-Word* dan *Document-Topic*

Variabel	Dokumen	Topik	Probabilitas
Left	1	1	$1 * 16.67 / (2 + (3 * 16.67)) = 0.3205$
		2	$1 * 16.67 / (2 + (3 * 16.67)) = 0.3205$
		3	$0 * 16.67 / (2 + (3 * 16.67)) = 0$

### III. ANALISIS DAN PERANCANGAN

---

Right	1	1	$1 * 0.1 / (2 + (4 * 0.1)) = 0.0416$
		2	$1 * 0.1 / (3 + (4 * 0.1)) = 0.0294$
		3	$0 * 0.1 / (1 + (4 * 0.1)) = 0$

10. Setelah mendapatkan probabilitas untuk *document-topic* (Left) dan *topic-word* (Right), langkah selanjutnya adalah menghitung nilai distribusi untuk semua topik dengan menyiapkan 2 variabel dimana variabel pertama (probZ) akan digunakan untuk menampung nilai hasil perkalian dari kedua probabilitas (Left dan Right) dan variabel kedua (sumProbZ) digunakan untuk mengakumulasi hasil perkalian dari kedua probabilitas (Left dan Right). Perhitungan probabilitas untuk variabel pertama (probZ) akan dibatasi dengan nilai  $\Lambda$ . Jika label bernilai 1 untuk topik tertentu maka nilai probabilitas = Left topik ke-k \* Right topik ke-k, sedangkan jika bernilai 0 maka nilai probabilitas = 0.

**Tabel 3.28** Contoh Perhitungan Distribusi Untuk Semua Topik

Variabel	Topik	$\Lambda$	Probabilitas
probZ	1	1	Left topik ke-1 * Right topik ke-1 = 0.01333
probZ	2	1	Left topik ke-2 * Right topik ke-2 = 0.00942
probZ	3	0	0
sumProbZ		$0.01333 + 0.00942 + 0 = 0.02275$	

11. Lakukan normalisasi variabel probZ untuk setiap topik yang ada. Normalisasi ini dapat dilakukan dengan cara membagi nilai probZ dengan nilai akhir sumProbZ ( $probZ / sumProbZ$ ).

**Tabel 3.29** Contoh Perhitungan Normalisasi Distribusi Untuk Semua Topik

Variabel	Topik	Probabilitas
probZ	1	$probZ \text{ topik } 1 / sumProbZ = 0.01333 / 0.02275 = 0.58593$
	2	$probZ \text{ topik } 2 / sumProbZ = 0.00942 / 0.02275 = 0.41406$
	3	$probZ \text{ topik } 3 / sumProbZ = 0 / 0.02275 = 0$

12. Lakukan *random sampling* dari nilai probZ untuk mendapatkan indeks topik yang baru. Untuk melakukan hal tersebut, siapkan variabel berjumlah K + 1. Variabel ke-1 digunakan untuk nilai sementara, sehingga nilainya adalah 0. Variabel ke-2 sampai ke-K merupakan

### III. ANALISIS DAN PERANCANGAN

---

nilai akumulasi probZ untuk setiap topik. Misalkan varibel tersebut bernama cumSumProbZ, maka hasil dari variabel ini akan menjadi seperti pada Tabel 3.30.

**Tabel 3.30** Contoh Perhitungan Akumulasi Nilai Normalisasi Distribusi Untuk Semua Topik

Variabel	Topik	Probabilitas
cumSumProbZ	0	0
	1	cumSumProbZ topik 0 + probZ topik 1 = 0 + 0.58593 = 0.58593
	2	cumSumProbZ topik 1 + probZ topik 2 = 0.58593 + 0.41406 = 0.99999
	3	cumSumProbZ topik 2 + probZ topik 3 = 0.99999 + 0 = 0.99999

13. Generate nilai *random* antara 0 sampai 1, lalu bandingkan nilai tersebut dengan nilai cumSumProbZ untuk semua topik dengan kondisi jika nilai random lebih besar sama dengan cumSumProbZ pada topik k dan nilai *random* lebih kecil cumSumProbZ pada topik k + 1 maka ambil topik k saat ini dan proses perbandingan selesai. Jika kondisi tersebut masih belum terpenuhi, bandingkan sampai kondisi tersebut terpenuhi atau semua nilai cumSumProbZ telah dibandingkan. Langkah ini merupakan langkah *Pseudocode* 2.3 pada baris 10.

Contoh:

Nilai *random* (*sample*) = 0.6

**Tabel 3.31** Contoh Proses Perbandingan Nilai Distribusi Setiap Topik

Topik	( <i>sample</i> $\geq$ cumSumProbZ topik k) dan ( <i>sample</i> $<$ cumSumProbZ topik k + 1)
1	$(0.6 \geq 0)$ dan $(0.6 < 0.58593)$ = salah
2	$(0.6 \geq 0.58593)$ dan $(0.6 < 0.99999)$ = benar
3	-

Berdasarkan perhitungan pada Tabel 3.31, proses perbandingan yang dilakukan hanya sampai topik ke-2, karena kondisi telah memenuhi. Setelah proses perbandingan selesai maka topik yang sesuai dengan kata “*classify*” ke-1 pada dokumen 1 adalah topik dengan indeks 2 yang berarti “ML”.

14. Ketika indeks topik yang baru telah didapat, maka tambahkan kembali (*increment*) jumlah kata pada matriks pada *document-topic* dan *topic-word* serta perbarui juga nilai pada matriks *topic assignment*. Setelah semua matriks diperbarui, lakukan hal yang sama untuk kata yang

### III. ANALISIS DAN PERANCANGAN

---

lainnya (kembali ke langkah 8). Langkah ini merupakan langkah *Pseudocode* 2.2 pada baris 9 dan 10 atau *Pseudocode* 2.3 pada baris 11 dan 12.

**Tabel 3.32** Contoh Penetapan Kembali Topik Untuk Kata ke-1 Dokumen 1

Topic Assignment		
	Kata-kata	Indeks Topik
<b>Doc 1</b>	classify	2
	recommend	2
	classify	1
<b>Doc 2</b>	image	3
	classify	2
<b>Doc 3</b>	text	2
	text	1

**Tabel 3.33** Contoh Pengamumsian Topik Untuk Kata ke-1 Dokumen 1

Topic-Word				
	classify	recommend	image	text
<b>NLP</b>	1	0	0	1
<b>ML</b>	2	1	0	1
<b>Citra</b>	0	0	1	0

**Tabel 3.34** Contoh Penambahan Kembali Jumlah Kata Matriks *Document-Topic*

Document-Topic			
	ML	NLP	Citra
<b>Doc 1</b>	1	2	0
<b>Doc 2</b>	0	1	1
<b>Doc 3</b>	1	1	0

15. Lakukan langkah 8-14 sebanyak kata yang ada dikalikan dengan jumlah iterasi. Dalam contoh ini maka lakukan sebanyak 7000 ( $7 * 1000$ ). Dalam *Pseudocode* 2.3, langkah ini merupakan langkah pada baris 1 sampai 3.

### III. ANALISIS DAN PERANCANGAN

---

16. Langkah terakhir adalah menghitung nilai  $\phi$  dengan menggunakan nilai-nilai yang ada pada matriks *topic-word* (Tabel 3.33). Untuk melakukan hal tersebut, setiap nilai pada matriks *topic-word* dikalikan dengan nilai  $\beta$ .

**Tabel 3.35** Contoh Hasil Matriks *Topic-Word* Dikalikan dengan  $\beta$

Topic-Word				
	classify	recommend	image	text
<b>NLP</b>	0.1	0	0	0.1
<b>ML</b>	0.2	0.1	0	0.1
<b>Citra</b>	0	0	0.1	0

17. Setelah semua nilai pada matriks *topic-word* dikalikan dengan  $\beta$ , jumlahkan nilai probabilitas semua topik untuk setiap kata yang ada. Lalu gunakan nilai tersebut untuk membagi nilai probabilitas yang ada pada matriks *topic-word* (jumlah kata w pada topik k / probabilitas kata w). Nilai pembagian antara probabilitas pada matriks *topic-word* dengan penjumlahan nilai probabilitas semua topik untuk setiap kata yang ada merupakan nilai  $\phi$  yang kemudian akan digunakan sebagai *classifier model*.

**Tabel 3.36** Contoh Proses Perbandingan Nilai Distribusi Setiap Topik

Kata	Jumlah Probabilitas
classify	$0.1 + 0.2 + 0 = 0.3$
recommend	$0 + 0.1 + 0 = 0.1$
image	$0 + 0 + 0.1 = 0.1$
text	$0.1 + 0.1 + 0 = 0.2$

**Tabel 3.37** Contoh Hasil Perhitungan Nilai  $\phi$

$\phi$				
	classify	recommend	image	text
<b>NLP</b>	0.333	0	0	0.5
<b>ML</b>	0.667	0.1	0	0.5
<b>Citra</b>	0	0	0.1	0

### 3.3.5 Classification

*Classification* merupakan bagian yang menjelaskan langkah untuk melakukan pengambilan keputusan. Sesuai dengan langkah SLLDA-C yang dijelaskan pada bab 2 bagian 2.1.10, langkah selanjutnya setelah mendapatkan nilai  $\phi$  adalah mengekstrak 20% kata dengan probabilitas tertinggi untuk setiap topik. Sebagai contoh, Tabel 3.38 merupakan 20% kata dengan probabilitas tertinggi untuk setiap topik.

**Tabel 3.38** Contoh 20% Kata Dengan Probabilitas Tertinggi Untuk Setiap Topik

Machine Learning	NLP	Image Processing
Latent	Text	Image
Train	Summarize	Object
Learn	Sentence	Edge
Classify	Token	Detect
Dataset	Information	Face
Automate	Dataset	Segment

Setelah memiliki kumpulan kata-kata dengan probabilitas tertinggi pada setiap topik, langkah terakhir SLLDA-C adalah menetapkan label ke dokumen yang diuji. Penetapan label dilakukan dengan melihat kata-kata yang paling banyak muncul pada suatu topik yang telah diekstrak. Berikut merupakan contoh dokumen yang diuji [4] (*machine learning* = garis bawah, *NLP* = huruf tebal, dan *image processing* = huruf miring).

*“Automatically categorizing news articles with high accuracy is an important task in an automated quick news system. We present two classifiers to classify news articles based on Labeled Latent Dirichlet Allocation, called LLDA-C and SLLDA-C. To verify classification accuracy we compare classification results obtained by the classifiers with those by trained professionals. We show that, through extensive experiments, both LLDA-C and SLLDA-C outperform SVM (Support Vector Machine, our baseline classifier) on precisions, particularly when only a small training dataset is available. SLLDA-C is also much more efficient than SVM. In terms of recalls, we show that LLDA-C is better than SVM. In terms of average Macro-F1 and Micro-F1 scores, we show that LLDA classifiers are superior over SVM. To further explore classifications of news articles we introduce the notion of content complexity, and study how content complexity would affect classifications.”*

Berdasarkan contoh dokumen di atas dengan menggunakan langkah SLLDA-C, dapat dilihat bahwa dokumen tersebut termasuk ke dalam kategori “*Machine Learning*”. Hal ini terjadi karena dalam dokumen tersebut kata-kata yang muncul pada kategori “*machine*

*learning*” lebih banyak dibandingkan jumlah kata yang muncul pada kategori lainnya. Jumlah kata yang muncul pada kategori “*Machine Learning*” adalah 16, sedangkan jumlah kata yang muncul untuk kategori “*NLP*” dan “*Image Processing*” adalah 1 dan 0. Dalam penelitian ini, keluaran dari dokumen yang diuji dapat menghasilkan lebih dari 1 kategori, sehingga untuk mendapatkan keluaran dengan lebih dari 1 kategori digunakan nilai *threshold* ketika melakukan pemilihan kategori yang sesuai. Nilai *threshold* yang digunakan adalah 40%. Dengan menggunakan nilai *threshold* yang telah ditetapkan, maka hasil untuk setiap kategori berdasarkan dokumen yang diuji akan seperti pada Tabel 3.39.

**Tabel 3.39** Contoh Pemilihan Kategori dengan Menggunakan Nilai *Threshold*

Topic	Jumlah	Persentase Kemunculan	Threshold $\geq 40\%$
Machine Learning	16	$16/17 = 0.9412 = 94.12\%$	Benar
NLP	1	$1/17 = 0.0588 = 5.88\%$	Salah
Image Processing	0	$0/17 = 0 = 0\%$	Salah

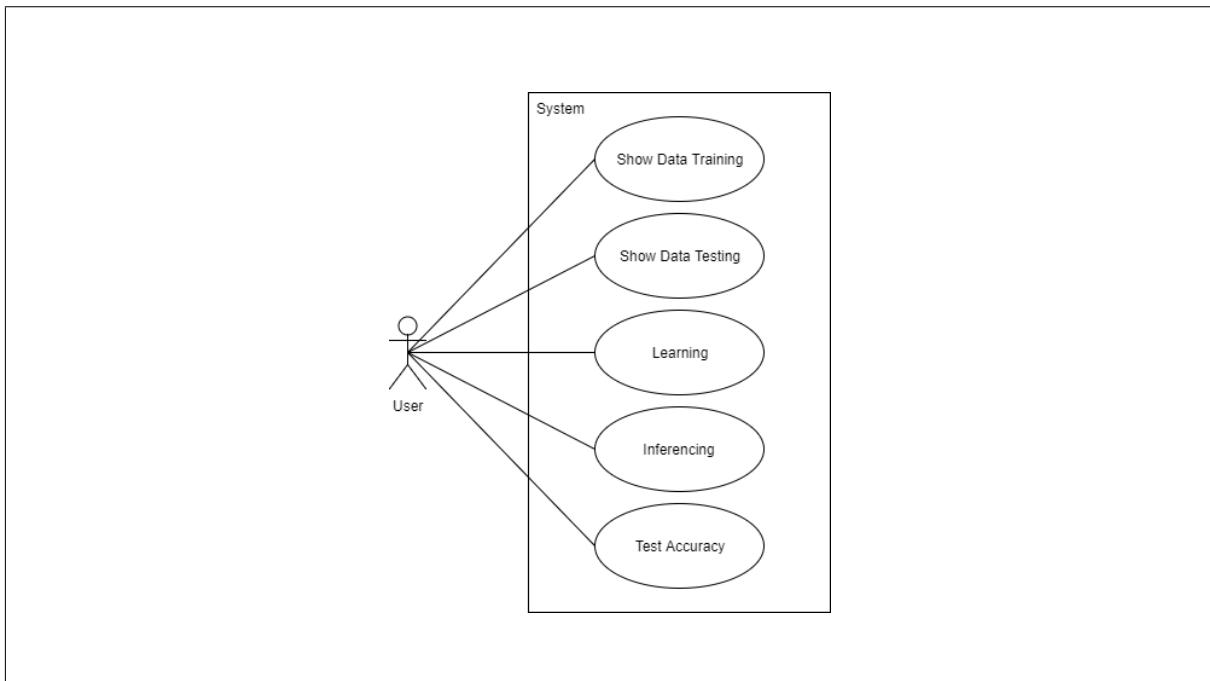
### 3.4 Tahap Perancangan Implementasi Sistem

Tahap perancangan implementasi sistem merupakan bagian yang menjelaskan mengenai *use case diagram* dan *class diagram* dari sistem yang akan dibangun.

#### 3.4.1 Use Case Diagram

Berikut merupakan fungsi-fungsi yang dapat dilakukan *user* ketika menggunakan sistem kategorisasi dokumen yang digambarkan dalam bentuk *use case*.

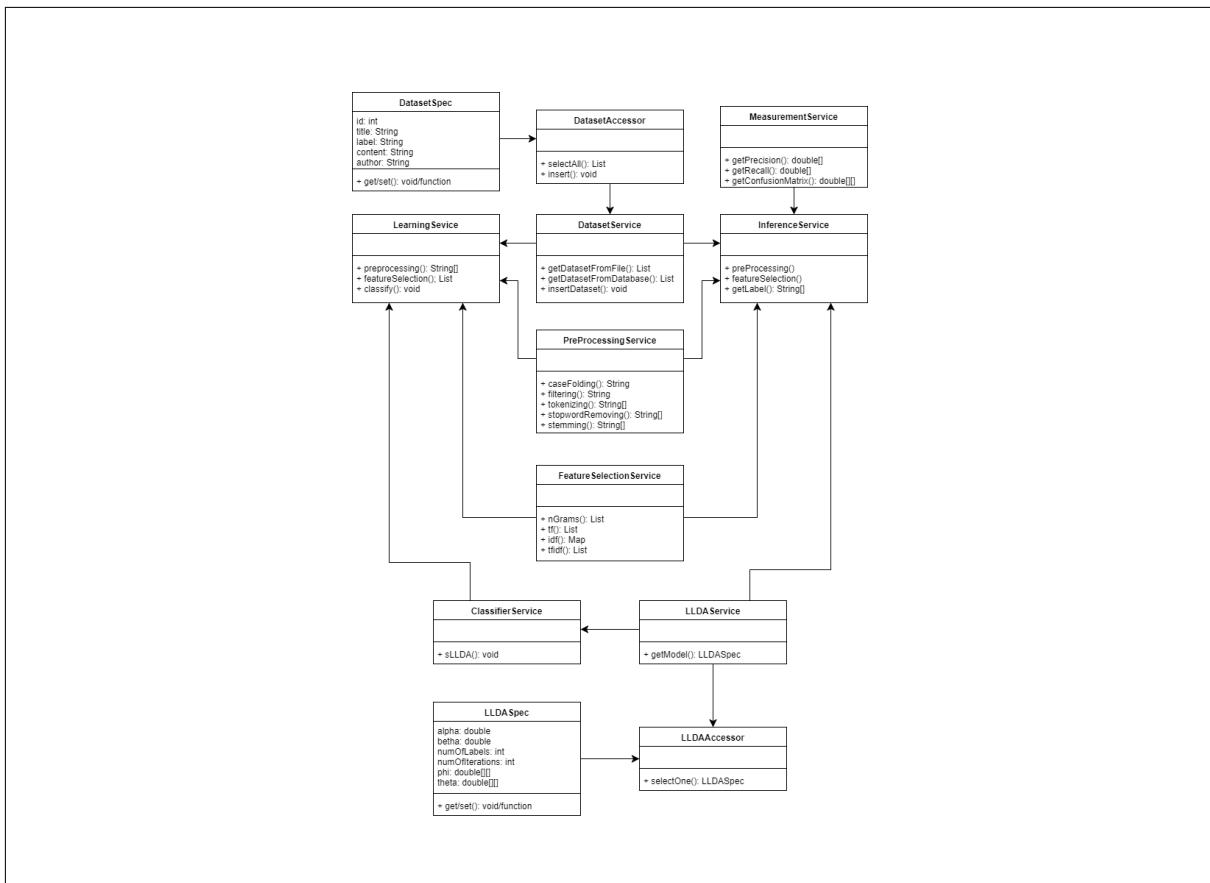
1. *Show Data Training*, merupakan fungsi untuk melihat daftar data belajar yang digunakan untuk proses pembelajaran.
2. *Show Data Testing*, merupakan fungsi untuk melihat daftar data uji yang digunakan untuk proses pengujian.
3. *Learning*, merupakan fungsi untuk melakukan pembelajaran sesuai dengan parameter yang diinginkan user.
4. *Inferencing*, merupakan fungsi untuk melakukan pengujian terhadap 1 dokumen yang ingin diuji. Pada fungsi ini, user dapat melihat hasil keluaran dari sistem yang berupa label.
5. *Test Accuracy*, merupakan fungsi untuk melihat nilai akurasi antara data belajar yang telah diproses dengan data uji.



**Gambar 3.5 Use Case Diagram**

### 3.4.2 Class Diagram

Berikut merupakan *class* dan *method* yang akan digunakan dalam pembangunan sistem kategorisasi dokumen yang digambarkan dalam bentuk *class diagram*.



**Gambar 3.6 Class Diagram**

## **BAB IV**

### **IMPLEMENTASI DAN PENGUJIAN**

#### **4.1 Lingkungan Implementasi**

Lingkup implementasi merupakan bagian yang menjelaskan mengenai spesifikasi perangkat keras dan perangkat lunak yang digunakan dalam membangun sebuah sistem.

##### **4.1.1 Spesifikasi Perangkat Keras**

Spesifikasi perangkat keras yang penulis gunakan dalam membangun sistem kategorisasi dokumen adalah sebagai berikut :

1. Laptop dengan processor Intel® Core™ i7-7500U CPU @ 2.70 GHz 2.90 GHz
2. RAM 8.00 GB
3. Hard disk 1 TB
4. AMD Radeon™ R7 M445

##### **4.1.2 Lingkungan Perangkat Lunak**

Spesifikasi perangkat lunak yang penulis gunakan dalam membangun sistem kategorisasi dokumen adalah sebagai berikut :

1. Sistem Operasi: Windows 10 64-bit
2. Tools Pengembangan: Java Development Kit 1.8.0 Update 64-bit, Netbeans IDE 8.0.2
3. DBMS: MySQL 5.0.12

#### **4.2 Implementasi Perangkat Lunak**

Implementasi perangkat lunak merupakan bagian yang menjelaskan mengenai implementasi sistem yang dibangun baik dalam bentuk *desktop* maupun *web*. Dalam penelitian ini, bentuk aplikasi yang dibangun adalah berbentuk *web*.

##### **4.2.1 Implementasi Basis Data**

Dalam pembangunan sistem kategorisasi dokumen, semua data seperti data untuk *training*, data untuk *testing*, dan hasil model pembelajaran akan disimpan dalam sebuah *database*. Tujuan penyimpanan data-data tersebut dalam sebuah *database* adalah untuk memungkinkan *user* untuk melakukan CRUD terhadap data *training* dan data *testing*, mencatat model setiap dilakukan pembelajaran dengan parameter yang berbeda, dan mempermudah pembacaan data. Jumlah tabel yang akan digunakan dalam pembangunan sistem adalah berjumlah 3 tabel. Tabel ini di antaranya adalah tabel data\_training, data\_testing, dan llda. Berikut merupakan struktur untuk setiap tabel yang digunakan.

## IV. IMPLEMENTASI DAN PENGUJIAN

---

### 1. Tabel data\_training

**Tabel 4.1** Struktur Tabel Data\_Training

Name	Type	Length	Null	Default	PK
id	int	11	No	None	1
title	varchar	200	No	None	-
label	Varchar	100	No	None	-
content	longtext	-	No	None	-
author	varchar	500	No	None	-
created_at	datetime	-	No	CURRENT_TIMESTAMP	-
deleted_at	datetime	-	Yes	Null	-

### 2. Tabel data\_testing

**Tabel 4.2** Struktur Tabel Data\_Testing

Name	Type	Length	Null	Default	PK
id	int	11	No	None	1
title	varchar	200	No	None	-
label	varchar	100	No	None	-
content	longtext	-	No	None	-
author	varchar	500	No	None	-
created_at	datetime	-	No	CURRENT_TIMESTAMP	-
deleted_at	datetime	-	Yes	Null	-

### 3. Tabel llda

**Tabel 4.3** Struktur Tabel LLDA

Name	Type	Length	Null	Default	PK
id	int	11	No	None	1
alpha	double	-	No	None	-
beta	double	-	No	None	-
n	int	11	No	1	-
d	int	11	No	None	-

k	int	11	No	None	-
iteration	int	-	No	None	-
vocabulary	longtext	-	No	None	-
phi	longtext	-	No	None	-
theta	longtext	-	No	None	-
created_at	datetime	-	No	CURRENT_TIMESTAMP	-

#### 4.2.2 Daftar Class dan Method

Daftar *Class* dan *Method* merupakan bagian yang menjelaskan tentang *Class* dan *Method* yang akan digunakan dalam membangun sistem kategorisasi dokumen. Dalam pembuatan sistem, terdapat 2 jenis *package* yang berfungsi untuk memisahkan *class* sesuai dengan fungsinya masing-masing. Kedua jenis *package* ini adalah *package final\_project\_api* dan *final\_project\_impl*. *Package final\_project\_api* berfungsi untuk menyimpan semua *interface* (*api*, *service*, dan *database*) dan data model, sedangkan *package final\_project\_impl* berfungsi untuk menyimpan semua *class implementation* berdasarkan *method* pada setiap *interface* yang telah didefinisikan pada *package final\_project\_api*. Di bawah ini merupakan daftar *class* untuk *package final\_project\_api* dan *final\_project\_impl*.

**Tabel 4.4** Daftar *Class* pada *Package final\_project\_api*

Package	Class	Jenis Class
final_project_api	DatasetAPI	<i>Interface</i>
	DatasetAccessor	<i>Interface</i>
	DatasetService	<i>Interface</i>
	DatasetSpec	<i>Class</i>
	PreProcessingService	<i>Interface</i>
	FeatureSelectionService	<i>Interface</i>
	classifierService	<i>Interface</i>
	LLDAccessor	<i>Interface</i>
	LLDASpec	<i>Class</i>
	LabelAPI	<i>Interface</i>
	LearningAPI	<i>Interface</i>
	LearningService	<i>Interface</i>

#### IV. IMPLEMENTASI DAN PENGUJIAN

---

	DatatestAPI	<i>Interface</i>
	DatatestAccessor	<i>Interface</i>
	DatatestService	<i>Interface</i>
	InferenceAPI	<i>Interface</i>
	InferenceService	<i>Interface</i>
	InferenceSpec	<i>Class</i>
	MeasurementAPI	<i>Interface</i>
	MeasurementService	<i>Interface</i>
	Label	<i>Enum</i>
	LabelConfig	<i>Enum</i>

**Tabel 4.5** Daftar *Class* pada *Package* final\_project\_impl

Package	Class
	DatasetAPIImpl
	DatasetAccessorImpl
	DatasetServiceImpl
	PreProcessingServiceImpl
	FeatureSelectionServiceImpl
	ClassifierServiceImpl
	LLDAccessorImpl
	LabelAPIImpl
	LearningAPIImpl
	LearningServiceImpl
	DatatestAPIImpl
	DatatestAccessorImpl
	DatatestServiceImpl
	InferenceAPIImpl
final_project_impl	

	InferenceServiceImpl
	MeasurementAPIImpl
	MeasurementServiceImpl

Dalam pembuatan *class*, terdapat 4 jenis *class* yang digunakan penulis dalam pembangunan sistem kategorisasi dokumen. Keempat jenis *class* ini di antaranya adalah *API*, *service*, *accessor*, dan *specification*. *Class API* berfungsi untuk menyediakan *method* untuk menjadi penghubung antara bagian *back-end* dan *front-end*. *Class service* berfungsi untuk menyelesaikan masalah yang dihadapi (*business logic*) serta menjadi penghubung antara *API* dengan *accessor*. *Class accessor* berfungsi untuk mengelola semua kegiatan yang berkaitan dengan *database*. *Class specification* berfungsi untuk mengelola data model selama pembangunan sistem berjalan.

#### 1. DatasetSpec.java

*Class DatasetSpec* merupakan *class* yang berfungsi sebagai data model dalam pengambilan data. *DatasetSpec* digunakan untuk memetakan data yang didapat dari *database* ke dalam sistem.

**Tabel 4.6** Daftar Atribut *Dataset Specification*

Tipe	Nama Atribut	Keterangan
int	id	Nomor id untuk setiap dokumen
String	title	Judul dokumen
String	label	Topik / Kategori dokumen
String	content	Isi dokumen
String	author	Pembuat dokumen

**Tabel 4.7** Daftar Fungsi dan Prosedur *Dataset Specification*

Tipe	Nama Atribut	Keterangan
void	set setiap element	Prosedur <i>setter</i> untuk setiap elemen atribut
function	get setiap element	Fungsi <i>getter</i> untuk setiap elemen atribut

#### 2. LLDApSpec.java

*Class LLDApSpec* merupakan *class* yang berfungsi sebagai data model dalam melakukan

#### IV. IMPLEMENTASI DAN PENGUJIAN

---

proses klasifikasi. Selain digunakan untuk proses klasifikasi, *class* ini juga berfungsi untuk memetakan data *llda* untuk disimpan ke *database* maupun diambil dari *database*.

**Tabel 4.8** Daftar Atribut LLDA Specification

Tipe	Nama Atribut	Keterangan
double	alpha	Nilai $\alpha$ untuk <i>llda</i>
double	beta	Nilai $\beta$ untuk <i>llda</i>
String	type	Jenis dokumen
int	n	Jumlah n pada n-gram yang mau digunakan
int	numOfLabels	Jumlah banyak kategori yang ada
int	numOfIterations	Jumlah iterasi
Map<String, Integer[]>	vocabularies	Kumpulan kata-kata yang telah diberi indeks
double[][]	phi	Nilai probabilitas topik terhadap kata
double[][]	theta	Nilai probabilitas dokumen terhadap topik

**Tabel 4.9** Daftar Fungsi dan Prosedur LLDA Specification

Tipe	Nama Atribut	Keterangan
void	set setiap element	Prosedur <i>setter</i> untuk setiap elemen atribut
function	get setiap element	Fungsi <i>getter</i> untuk setiap elemen atribut

#### 3. DatasetAccessorImpl.java

*Class* DatasetAccessorImpl merupakan *class* yang digunakan untuk melakukan proses CRUD (*create*, *read*, *update*, dan *delete*) terhadap data *training* ke *database*.

**Tabel 4.10** Daftar Fungsi dan Prosedur Dataset Accessor

Tipe	Nama Atribut	Keterangan
void	setDataSource	Prosedur untuk menghubungkan sistem dengan tabel yang ada pada <i>database</i>
List<DatasetSpec>	selectAll	Fungsi untuk mendapatkan data <i>training</i> dari <i>database</i>
void	insert	Prosedur untuk menambahkan data <i>training</i> ke <i>database</i>

#### 4. DatasetServiceImpl.java

*Class DatasetServiceImpl merupakan class yang digunakan untuk mengolah data training yang didapat dari database.*

**Tabel 4.11** Daftar Fungsi dan Prosedur *Dataset Service*

Tipe	Nama Atribut	Keterangan
List<DatasetSpec>	getDatasetFromFile	Fungsi untuk mendapatkan data <i>training</i> dari sebuah <i>file</i>
List<DatasetSpec>	getDatasetFromDB	Fungsi untuk mendapatkan data <i>training</i> dari <i>database</i>
void	insertDataset	Prosedur untuk menambahkan data <i>training</i> ke <i>database</i>

#### 5. PreProcessingServiceImpl.java

*Class PreProcessingServiceImpl adalah class yang digunakan untuk mengolah raw document menjadi clean document.*

**Tabel 4.12** Daftar Fungsi dan Prosedur *Pre-Processing Service*

Tipe	Nama Atribut	Keterangan
String	caseFolding	Fungsi untuk mengubah bentuk huruf pada isi dokumen menjadi huruf kecil
String	filtering	Fungsi untuk menghapus tanda baca dan kata penghubung
String[]	tokenizing	Fungsi untuk memecah suatu isi dokumen menjadi kata-kata
String[]	stopWordRemoving	Fungsi untuk menghapus makna yang kurang bermakna
String[]	stemming	Fungsi untuk mengubah setiap kata menjadi kata dasar pembentuknya

#### 6. FeatureSelectionServiceImpl.java

*Class FeatureSelectionServiceImpl merupakan class yang digunakan untuk menyeleksi kembali kata-kata yang telah melalui pre-processing.*

## IV. IMPLEMENTASI DAN PENGUJIAN

---

**Tabel 4.13** Daftar Fungsi dan Prosedur *Feature Selection Service*

<b>Tipe</b>	<b>Nama Atribut</b>	<b>Keterangan</b>
List<String[]>	nGrams	Fungsi untuk mengubah kata-kata menjadi n-gram
List<Map<String, Double>>	tf	Fungsi untuk menghitung nilai tf
Map<String, Double>	Idf	Fungsi untuk menghitung nilai idf
List<String[]>	tfidf	Fungsi untuk menghitung nilai tf-idf

### 7. ClassifierServiceImpl.java

Class ClassifierServiceImpl merupakan *class* yang digunakan untuk melakukan klasifikasi terhadap dokumen. Dalam penelitian ini, metode klasifikasi yang digunakan adalah menggunakan *Simple Labeled Latent Dirichlet Allocation* (sLLDA).

**Tabel 4.14** Daftar Fungsi dan Prosedur *Classifier Service*

<b>Tipe</b>	<b>Nama Atribut</b>	<b>Keterangan</b>
void	sLLDA	Prosedur untuk melakukan klasifikasi dengan sLLDA
List<double[]>	initializeΛ	Fungsi untuk melakukan inisialisasi nilai $\Lambda$
Map<String, Integer[]>	initializeVocabularies	Fungsi untuk melakukan inisialisasi kata-kata
double[][]	initializeMatrix	Fungsi untuk melakukan inisialisasi setiap matriks
void	generateTopicOverWords	Prosedur untuk mendapatkan jumlah setiap kata pada setiap topik
void	generateDocOverTopics	Prosedur untuk mendapatkan jumlah setiap dokumen pada setiap topik
void	gibbsSampling	Prosedur untuk melakukan pembelajaran dengan <i>Gibbs Sampling</i>
double[][]	computePhi	Fungsi untuk menghitung nilai $\phi$
double[][]	computeTheta	Fungsi untuk menghitung nilai $\theta$

### 8. LLDAAccessorImpl.java

#### IV. IMPLEMENTASI DAN PENGUJIAN

---

*Class LLDAAccessorImpl* merupakan *class* yang digunakan untuk melakukan proses *read* dan *insert* model pembelajaran ke *database*.

**Tabel 4.15** Daftar Fungsi dan Prosedur *LLDA Accessor*

Tipe	Nama Atribut	Keterangan
void	setDataSource	Prosedur untuk menghubungkan sistem dengan tabel yang ada pada <i>database</i>
LLDASpec	selectOne	Fungsi untuk mendapatkan model dari <i>database</i>
void	insert	Fungsi untuk menambah model baru ke <i>database</i>

#### 9. LearningServiceImpl.java

*Class LearningServiceImpl* merupakan *class* yang digunakan untuk mengolah data *training* mulai dari pengolahan data sampai menjadi suatu model.

**Tabel 4.16** Daftar Fungsi dan Prosedur *Learning Service*

Tipe	Nama Atribut	Keterangan
String[]	preProcessing	Fungsi untuk melakukan <i>pre-processing</i>
List<String[]>	featureSelection	Fungsi untuk menyeleksi kata-kata yang akan digunakan
void	classify	Prosedur untuk melakukan klasifikasi

#### 10. InferenceServiceImpl.java

*Class InferenceServiceImpl* merupakan *class* yang digunakan untuk melakukan proses pengujian terhadap dokumen yang diuji.

**Tabel 4.17** Daftar Fungsi dan Prosedur *Inference Service*

Tipe	Nama Atribut	Keterangan
String[]	preProcessing	Fungsi untuk melakukan <i>pre-processing</i>
String[]	featureSelection	Fungsi untuk mengubah bentuk kata menjadi n-gram
List<List<InferenceSpec>>	getModel	Fungsi untuk mendapatkan model pembelajaran
String[]	getLabels	Fungsi untuk mendapatkan hasil label

#### 11. MeasurementServiceImpl.java

*Class MeasurementServiceImpl* merupakan *class* yang digunakan untuk menghitung nilai

*precision* dan *recall*.

**Tabel 4.18** Daftar Fungsi dan Prosedur *Measurement Service*

Tipe	Nama Atribut	Keterangan
void	calculateMatrix	Prosedur untuk menghitung jumlah label yang sebenarnya terhadap label yang diprediksi
double[]	getPrecision	Fungsi untuk menghitung nilai <i>precision</i>
double[]	getRecall	Fungsi untuk menghitung nilai <i>recall</i>
double	getTP	Fungsi untuk mendapatkan nilai <i>true positive</i>
double	getFN	Fungsi untuk mendapatkan nilai <i>false negative</i>
double	getFP	Fungsi untuk mendapatkan nilai <i>false positive</i>

#### 4.2.3 Implementasi Struktur Data

Implementasi Struktur Data merupakan bagian yang menjelaskan mengenai langkah-langkah setiap proses yang terdapat dalam sistem kategorisasi dokumen. Langkah yang akan dijelaskan adalah langkah mulai dari masukan sampai dengan keluaran. Langkah setiap proses akan direpresentasikan dalam bentuk *pseudocode* agar lebih mudah untuk dipahami.

##### 1. Pengambilan data

- (a) Untuk pertama kali, lakukan *insert* data belajar ke *database* dengan cara membaca data belajar dari *file .xls*.
- (b) Melakukan pengambilan data dari *database*. Data yang diambil dari *database* akan ditampung dalam tipe data *List<DatasetSpec>*. *List* menunjukkan kumpulan dokumen dan *DatasetSpec* merupakan *object dataset* yang berisi *title*, *label*, *content*, dan *author*.

##### 2. *Pre-Processing*

- (a) Melakukan *case folding* dengan menggunakan fungsi *.toLowerCase()*
- (b) Melakukan *filtering* dengan menggunakan fungsi *.replaceAll(x)*, dimana parameter dari fungsi tersebut merupakan *regular expression*.
- (c) Melakukan *tokenizing* dengan menggunakan fungsi *.split(" ")*
- (d) Melakukan *stopword removing* dengan cara mencocokan daftar *stopword* yang ditampung dalam tipe data *Set<String>* terhadap kata-kata yang terdapat dalam suatu dokumen menggunakan fungsi *.contains(word)*.

- (e) Melakukan *stemming* menggunakan *library snowball stemmer*. Fungsi yang dapat digunakan untuk melakukan *stemming* adalah .stem().

#### 3. Feature Selection

- (a) Melakukan pembentukan kata (n-grams) berdasarkan nilai n yang dimasukan. Pembentukan kata ini dapat berupa *unigram*, *bigram*, atau *trigram*.
- (b) Menghitung nilai *term frequency* dengan menampung nilai tersebut dalam tipe data Map<String, Double>. *Key* dari tipe data tersebut merupakan kata dari suatu dokumen dan *value* merupakan jumlah kata pada suatu dokumen yang telah dinormalisasi.
- (c) Menghitung nilai *inverse document frequency* dengan menggunakan fungsi Math.log10(N/df) dan menampung nilai tersebut dalam tipe data Map<String, Double>. *Key* dari tipe data tersebut merupakan kata dari semua dokumen dan *value* merupakan nilai idf untuk setiap katanya.
- (d) Menghitung nilai TF-IDF dengan cara mengalikan nilai tf dengan nilai idf yang telah didapat pada langkah sebelumnya. Hasil dari TF-IDF ditampung dalam tipe data Map<String, Double>. *Key* dari tipe data tersebut merupakan kata dari semua dokumen dan *value* merupakan nilai tf-idf untuk setiap katanya.
- (e) Lakukan pemilihan n TF-IDF tertinggi. Nilai n pada langkah ini dapat bernilai 100, 200, 300, 400, 500, dst.
- (f) Setelah mendapat n TF-IDF tertinggi, lakukan pemilihan kata sesuai dengan kata-kata dengan nilai TF-IDF tertinggi terhadap kata-kata pada dokumen yang telah melalui tahap *pre-processing*. Hasil dari pemilihan kata ini akan ditampung dalam tipe data List<String[]>. *List* menunjukkan kumpulan dokumen dan String[] menunjukkan kumpulan kata-kata.

#### 4. Learning

- (a) Melakukan inisialisasi untuk nilai  $\Lambda$ . Nilai  $\Lambda$  akan ditampung dalam tipe data List<double[]>. *List* menunjukkan kumpulan dokumen dan double[] menunjukkan indikator ada tidaknya label (1 atau 0).
- (b) Melakukan *indexing* terhadap semua kata-kata yang telah melalui tahap *pre-processing* dan *feature selection*. Hasil indexing akan ditampung dalam tipe data Map<String, Integer[]>. *Key* dari tipe data tersebut merupakan kata dari semua dokumen dan Integer[] menunjukkan indeks kata dan jumlah suatu kata pada semua dokumen.

- (c) Melakukan inisialisasi terhadap matriks *topic-word*, *document-topic*, dan *topic assignment*. Tipe data untuk matriks *topic-word* dan *document-topic* adalah *double[][]*, sedangkan tipe data untuk matriks *topic assignment* adalah *List<double[]>*. List pada matriks *topic assignment* menunjukkan kumpulan dokumen dan *double[]* menunjukkan indeks topik untuk setiap kata.
- (c) Melakukan generalisasi terhadap matriks *topic-word*. Generalisasi ini bertujuan untuk menetapkan topik awal untuk setiap kata yang ada pada semua dokumen. Dalam langkah ini, nilai pada matriks *topic-word* dan *topic assignment* akan dilakukan *increment* pada kolom dan baris yang sesuai.
- (c) Melakukan generalisasi terhadap matriks *document-topic*. Generalisasi ini bertujuan untuk melakukan *increment* untuk nilai pada matriks *document-topic*.
- (c) Melakukan perbaruan topik untuk setiap kata pada seluruh dokumen. Perbaruan ini bertujuan agar setiap topik yang ditetap untuk setiap kata menjadi lebih sesuai. Langkah ini dapat dilakukan dengan menggunakan algoritma *Gibbs Sampling*. Pada algoritma ini, setiap kata akan dilakukan perulangan sebanyak jumlah iterasi yang telah ditetapkan (contoh: 1000). Dalam iterasi akan dilakukan perhitungan *multinomial distribution* untuk probabilitas *topic-word* dan *document-topic*. Perhitungan tersebut akan digunakan untuk basis penetapan topik baru.
- (c) Menghitung nilai  $\phi$  (model) yang didapat dari matriks *topic-word* yang dikalikan dengan beta kemudian dilakukan normalisasi. Nilai  $\phi$  akan ditampung dalam tipe data *double[][]*.

### 5. Inference

- (a) Ekstrak 20% kata dengan probabilitas tertinggi untuk setiap topik berdasarkan model ( $\phi$ ) yang telah didapat pada langkah *learning*.
- (b) Melakukan pencocokan antara kata-kata pada dokumen yang diuji dengan kata-kata yang telah diekstrak. Lakukan penambahan (increment) pada topik yang sesuai jika hasil pencocokan sama.
- (c) Jika kasus *single label*, ambil indeks untuk jumlah topik tertinggi dan keluarkan hasil topiknya.
- (d) Jika kasus *multi label*, hitung persentase kemunculan untuk semua topik dan bandingkan persentase kemunculan tersebut dengan nilai *threshold* 40%. Ketika sudah dibandingkan keluarkan hasil topiknya.

## 4.3 Pengujian

Pengujian merupakan bagian yang menjelaskan hasil keluaran dari sistem kategorisasi dokumen terhadap dokumen penelitian yang diuji.

### 4.3.1 Hasil Pengujian

Dalam penelitian ini, pengujian akan dilakukan dalam beberapa kasus. Kasus yang akan diuji di antaranya adalah pengujian *unigram* tanpa TF-IDF dengan n TF-IDF terbaik dan pengujian *bag of words* (*bigram* dan *trigram*) tanpa TF-IDF dengan n TF-IDF terbaik yang didapat pada kasus *unigram*. Hasil pengujian yang akan diuji terhadap kasus di atas adalah tingkat akurasi. Dalam proses pengujian nilai  $\alpha$ ,  $\beta$ , jumlah dokumen, jumlah topik, dan jumlah iterasi pada model pembelajaran yang dilakukan akan bernilai sama, sedangkan untuk nilai  $\phi$  akan berbeda. Berikut merupakan nilai  $\alpha$ ,  $\beta$ , jumlah topik (K), jumlah iterasi, jumlah data *training*, dan jumlah data *testing* untuk setiap pengujian.

**Tabel 4.19** Parameter Pengujian

<b><math>\alpha</math></b>	<b><math>\beta</math></b>	<b>K</b>	<b>Iterasi</b>	<b>Data Training</b>	<b>Data Testing</b>
16.667	0.1	3	1000	180	20

#### 4.3.1.1 Pengujian *Unigram* Tanpa TF-IDF dengan n TF-IDF

Dalam pengujian ini akan dilakukan perbandingan hasil keluaran dari sistem yang berupa *single label* maupun *multi label* dengan label asli dari data yang diuji. Kondisi perbandingan akan dilakukan dengan melihat rata-rata *F-Measure* menggunakan n TF-IDF terbaik dengan tanpa menggunakan TF-IDF. Jumlah TF-IDF terbaik yang akan digunakan dalam pengujian ini, akan diuji terlebih dahulu untuk mengetahui saat n ke berapa sistem dapat mengeluarkan hasil rata-rata *F-Measure* tertinggi. Batasan untuk jumlah n yang akan diuji adalah 1000 buah dengan jarak 100. Jenis *bag of words* yang akan diuji adalah *unigram*. Berikut merupakan hasil beserta analisis dari pengujian yang telah dilakukan.

##### 1. Single Label

**Tabel 4.20** Pengujian *Unigram* Tanpa TF-IDF Single Label

	<b>Machine Learning</b>	<b>NLP</b>	<b>Image Processing</b>	<b>Rata-Rata</b>
<i>Unigram</i> , tanpa TF-IDF, <i>single label</i>	0.933333333	1	0.923076923	0.952136752

**Tabel 4.21** Pengujian *Unigram n* TF-IDF Single Label

<b>n TF-IDF</b>	<b>Machine Learning</b>	<b>NLP</b>	<b>Image Processing</b>	<b>Rata-Rata</b>
100	0.615384615	0.75	0.769230769	0.711538462
200	0.888888889	0.8	1	0.896296296
300	0.75	0.666666667	1	0.805555556
400	0.777777778	0.727272727	0.923076923	0.809375809
500	0.842105263	0.666666667	1	0.83625731
<b>600</b>	<b>0.875</b>	<b>0.833333333</b>	<b>1</b>	<b>0.902777778</b>
700	0.8	0.714285714	0.909090909	0.807792208
800	0.833333333	0.823529412	0.909090909	0.855317885
900	0.8	0.714285714	0.909090909	0.807792208
1000	0.8	0.823529412	0.923076923	0.848868778

Berdasarkan hasil pengujian di atas, hasil rata-rata *F-Measure* terbaik untuk kasus *n* TF-IDF *single label* terjadi ketika TF-IDF yang digunakan berjumlah 600 buah. Jika hasil rata-rata pengujian *F-Measure* untuk kasus tanpa TF-IDF dan 600 TF-IDF tertinggi dibandingkan, hasil menggunakan 600 TF-IDF tertinggi lebih kecil dibanding hasil rata-rata tanpa menggunakan TF-IDF. Perbedaan hasil rata-rata dari kedua pengujian ini sebesar 0.05 atau jika diubah ke dalam persentase perbedaannya adalah 5%. Jika dilihat dari hasil *F-Measure* untuk setiap topik, *F-Measure* untuk topik “*Image Processing*” pada kasus 600 TF-IDF lebih besar dibanding pada kasus tanpa TF-IDF, sedangkan untuk topik “*Machine Learning*” dan “*Natural Language Processing*” hasil *F-Measure* pada kasus 600 TF-IDF lebih kecil.

Jika dianalisis lebih lanjut berdasarkan *confusion matrix* yang dihasilkan oleh kedua kasus tersebut, kasus tanpa TF-IDF menghasilkan *F-Measure* yang lebih besar karena jumlah kesalahan yang dihasilkan oleh sistem terhadap dokumen yang diuji hanya berjumlah 1, sedangkan untuk 600 TF-IDF tertinggi jumlah kesalahan yang dihasilkan adalah berjumlah 2 dokumen. Berikut merupakan *confusion matrix* untuk kedua kasus tersebut.

#### IV. IMPLEMENTASI DAN PENGUJIAN

---

**Tabel 4.22** Confusion Matrix Unigram Tanpa TF-IDF Single Label

	Machine Learning	NLP	Image Processing
Machine Learning	7	0	1
NLP	0	6	0
Image Processing	0	0	6

**Tabel 4.23** Confusion Matrix Unigram n TF-IDF Single Label

	Machine Learning	NLP	Image Processing
Machine Learning	7	1	0
NLP	1	5	0
Image Processing	0	0	6

Kesalahan prediksi untuk kedua kasus tersebut terjadi pada dokumen yang berbeda. Untuk kasus tanpa TF-IDF, kesalahan terjadi pada dokumen berjudul “*A Group-Based Image Inpainting Using Patch Refinement in MRF Framework*” yang merupakan jurnal mengenai *Image Processing*, sedangkan untuk kasus 600 TF-IDF, kesalahan terjadi pada dokumen berjudul “*Text Classification Using Machine Learning*” yang merupakan jurnal mengenai *Machine Learning* dan dokumen berjudul “*An Automatic Text Summarization using Text Features and Singular Value Decomposition for Popular Articles in Indonesia Language*” yang merupakan jurnal penelitian mengenai *Natural Language Processing*. Analisis terhadap pengaruh penggunaan TF-IDF dan tanpa menggunakan TF-IDF akan dijelaskan pada bagian evaluasi (4.3.2).

#### 2. Multi Label

**Tabel 4.24** Pengujian Unigram tanpa TF-IDF Multi Label

	Machine Learning	NLP	Image Processing	Rata-Rata
Unigram, tanpa TF-IDF, multi label	0.592592593	0.631578947	0.777777778	0.667316439

**Tabel 4.25** Pengujian *Unigram* n TF-IDF *Multi Label*

<b>n TF-IDF</b>	<b>Machine Learning</b>	<b>NLP</b>	<b>Image Processing</b>	<b>Rata-Rata</b>
100	0.380952381	0.52173913	0.625	0.509230504
200	0.642857	0.555556	0.857143	0.685185
300	0.482758621	0.533333333	0.8	0.605363985
400	0.518518519	0.5	0.8	0.60617284
500	0.6	0.52173913	0.8	0.64057971
600	0.533333333	0.518518519	0.736842105	0.596231319
700	0.5	0.571428571	0.75	0.607142857
800	0.5	0.56	0.769230769	0.60974359
900	0.5	0.583333333	0.75	0.611111111
1000	0.416666667	0.538461538	0.666666667	0.540598291

Berdasarkan hasil pengujian di atas, hasil rata-rata *F-Measure* terbaik untuk kasus n TF-IDF *multi label* terjadi ketika TF-IDF yang digunakan berjumlah 200 buah. Jika hasil rata-rata pengujian *F-Measure* untuk kasus tanpa TF-IDF dan 200 TF-IDF tertinggi dibandingkan, hasil menggunakan 200 TF-IDF tertinggi lebih besar dibanding hasil rata-rata tanpa menggunakan TF-IDF. Perbedaan hasil rata-rata dari kedua pengujian ini sebesar 0.02 atau jika diubah ke dalam persentase perbedaannya adalah 2%.

Jika hasil *F-Measure* pada pengujian *multi label* dibandingkan dengan pengujian *single label*, perbedaan hasil rata-rata *F-Measure* dapat dikatakan besar. Rata-rata *F-Measure* untuk *multi label* berkisar antara 50% hingga 68%, sedangkan untuk *single label* nilai rata-rata *F-Measure* berkisar antara 80% hingga 95%. Hal ini dapat terjadi karena untuk kasus *multi label*, label yang dicocokan akan berjumlah lebih dari 1. Sebagai contoh jika keluaran dari sistem merupakan *multi label* dan jenis label pada data uji adalah *single label*, maka label pada data uji akan dilakukan pemeriksaan terhadap semua label yang dihasilkan oleh sistem. Hal ini akan mengakibatkan penambahan nilai untuk nilai *false positive*, sehingga akurasi yang dihasilkan dapat menurun.

#### 4.3.1.2 Pengujian *Bag of Words* Tanpa TF-IDF dengan n TF-IDF

Dalam pengujian ini, *bag of words* yang akan diuji adalah *bigram* dan *trigram*. Jumlah TF-IDF terbaik yang akan diuji dalam pengujian ini adalah jumlah TF-IDF terbaik yang didapatkan pada pengujian TF-IDF untuk *unigram*.

**Tabel 4.26** Pengujian *Bigram* dan *Trigram*

	<b>Machine Learning</b>	<b>NLP</b>	<b>Image Processing</b>	<b>Rata-Rata</b>
Bigram, Tanpa TF-IDF, <i>single label</i>	0.923076923	0.933333333	1	0.952136752
Bigram, Tanpa TF-IDF, <i>multi label</i>	0.545454545	0.666666667	0.823529412	0.678550208
Bigram, 600 TF-IDF, <i>single label</i>	0.434782609	0.476190476	0	0.303657695
Bigram, 200 TF-IDF, <i>multi label</i>	0.4	0.4	0	0.266666667
Trigram, Tanpa TF-IDF, <i>single label</i>	0.545454545	0.333333333	0.333333333	0.404040404
Trigram, Tanpa TF-IDF, <i>multi label</i>	0.4375	0.260869565	0.347826087	0.348731884
Trigram, 600 TF-IDF, <i>single label</i>	0.266666667	0.24	0	0.168888889
Trigram, 200 TF-IDF, <i>multi label</i>	0.1875	0.258064516	0	0.148521505

Berdasarkan hasil pengujian di atas, hasil rata-rata *F-Measure* terbaik untuk penggunaan *bigram* dan *trigram* adalah ketika jenis *bag of words* yang digunakan adalah *bigram*, tanpa menggunakan TF-IDF, dan keluaran dari sistem berupa *single label*. Jika hasil rata-rata untuk setiap kasus dianalisis, hasil *bigram* dengan TF-IDF dan hasil *trigram* untuk semua kasus memiliki nilai akurasi yang sangat rendah. Untuk kasus *bigram*, hal ini terjadi karena jumlah teks yang terdiri dari 2 kata yang sama akan berjumlah sedikit. Dalam pengujian ini, jumlah teks yang terdiri dari 2 kata yang sama sebagian besar berjumlah 1 dan 2. Jumlah kemunculan kata yang sedikit tersebut dapat menyebabkan model yang dihasilkan menjadi tidak tepat. Dalam perhitungan *Labeled Latent Dirichlet Allocation*, model klasifikasi didapatkan dari hasil kali pada jumlah kata w pada topik yang sesuai dibagi dengan jumlah kata w pada seluruh topik. Jika kata w hanya ada 1 diantara semua dokumen, hasil dari model probabilitas yang dihasilkan untuk kata tersebut akan mendekati angka 1. Jika hal ini terjadi untuk semua kata, maka semua kata tersebut akan memiliki probabilitas mendekati angka 1.

Permasalahan yang akan terjadi ketika menggunakan *Simplified Labeled Latent Dirichlet Allocation* adalah metode tersebut akan mengekstrak 20% probabilitas kata tertinggi dari model yang telah diolah untuk dijadikan basis dalam pengambilan keputusan. Dengan melakukan ekstraksi ini, tidak semua kata-kata akan terpilih untuk dijadikan basis meskipun probabilitas untuk kata-katanya mendekati angka 1. Masalah tersebut berlaku juga untuk kasus *trigram*. Lalu permasalahan untuk akurasi *trigram* yang sangat kecil, disebabkan karena saat melakukan pengecekan antara teks dengan 3 suku kata pada dokumen yang diuji dengan teks pada model yang diekstrak banyak yang tidak cocok. Menurut hasil analisis yang didapat berdasarkan pengujian, kemunculan untuk cocoknya kata yang diuji dengan model yang diekstrak menggunakan *bigram* lebih tinggi persentase kemunculannya dibanding menggunakan *trigram*.

#### 4.3.2 Pengujian Waktu

Dalam pengujian ini akan dilakukan pengujian waktu komputasi saat melakukan proses pembelajaran terhadap pengujian terbaik yang dilakukan pada bagian 4.3.1. Pengujian yang akan diuji waktu komputasinya adalah penggunaan *unigram* tanpa TF-IDF, *unigram* menggunakan 600 TF-IDF tertinggi, *bigram* tanpa TF-IDF, dan *trigram* tanpa TF-IDF.

**Tabel 4.27** Pengujian Waktu Proses Pembelajaran

Fitur	Waktu
Unigram + Tanpa TF-IDF	1 menit 30 detik
Unigram + 600 TF-IDF	27 detik
Bigram + Tanpa TF-IDF	4 menit 30 detik
Trigram + Tanpa TF-IDF	5 menit 5 detik

Berdasarkan pengujian di atas, waktu pembelajaran tercepat terjadi ketika *Labeled Latent Dirichlet Allocation* menggunakan *unigram* dan 600 TF-IDF tertinggi. Hal ini dapat terjadi karena dengan memilih 600 TF-IDF tertinggi, jumlah kata yang akan digunakan dalam proses pembelajaran akan berkurang. Dengan berkurangnya jumlah kata akibat pemilihan 600 TF-IDF tertinggi, waktu komputasi untuk pembelajaran pun akan ikut berkurang juga.

#### 4.3.3 Evaluasi

Berdasarkan hasil pengujian yang telah dilakukan terhadap pengujian *unigram* tanpa TF-IDF dengan n TF-IDF terbaik dan pengujian *bag of words* (*bigram* dan *trigram*) tanpa TF-IDF dengan n TF-IDF terbaik yang didapat pada kasus *unigram*, hasil akurasi terbaik terjadi ketika klasifikasi *Labeled Latent Dirichlet Allocation* tidak menggunakan TF-IDF, jenis *bag of words* yang digunakan adalah *unigram* atau *bigram*, dan keluaran dari sistem merupakan *single label* dengan tingkat akurasi 95%. Hal ini dapat terjadi karena jika kata-kata

pada setiap dokumen digunakan semua, maka kecocokan antara dokumen yang diuji dengan dokumen yang digunakan untuk pembelajaran akan semakin mirip. Dalam TF-IDF, suatu kata akan bernilai tinggi jika kata tersebut hanya terdapat di beberapa dokumen saja dan jumlah kemunculan kata pada suatu dokumen berjumlah besar. Akibat kondisi tersebut, terdapat beberapa kata yang sebenarnya mewakili suatu topik tetapi kata-kata tersebut tidak terpilih karena tersebar di banyak dokumen. Sebagai contoh, Tabel 4.28 merupakan contoh *document frequency* untuk 5 dokumen yang telah memiliki label.

**Tabel 4.28** Contoh Permasalahan Menggunakan TF-IDF

	D1 (ML)	D2 (ML)	D3 (NLP)	D4 (ML)	D5 (NLP)	df
latent	1	1	1	1	0	4
sentence	0	0	1	0	0	1
word	0	0	0	0	1	1
classify	0	1	0	0	0	1

Berdasarkan contoh pada Tabel 4.28, kata “*latent*” muncul di 4 dokumen sedangkan kata “*sentence*”, “*word*”, dan “*classify*” hanya muncul di 1 dokumen saja. Jika TF-IDF digunakan untuk kondisi seperti ini, nilai untuk kata “*sentence*”, “*word*”, dan “*classify*” akan bernilai besar, sedangkan untuk kata “*latent*” akan bernilai kecil. Kata “*latent*” pada kasus ini sebenarnya dapat menggambarkan topik ML, karena jika menggunakan metode *Labeled Latent Dirichlet Allocation* probabilitas kata pada suatu dokumen akan dibatasi dengan suatu label yang dimiliki dokumen tersebut. Dengan kata lain, kata “*latent*” memungkinkan untuk masuk ke dalam topik ML. *Labeled Latent Dirichlet Allocation* adalah salah satu metode yang proses klasifikasinya berdasarkan kata-kata, sehingga jika kata-kata yang digunakan dalam sebuah dokumen berkurang, maka akan berpengaruh terhadap hasil klasifikasinya. Berikut merupakan *confusion matrix* untuk kasus *unigram* dan *bigram* yang terbaik.

**Tabel 4.29** Confusion Matrix Unigram Terbaik

	Machine Learning	NLP	Image Processing
Machine Learning	7	0	1
NLP	0	6	0
Image Processing	0	0	6

**Tabel 4.30** Confusion Matrix Bigram Terbaik

	<b>Machine Learning</b>	<b>NLP</b>	<b>Image Processing</b>
<b>Machine Learning</b>	6	0	0
<b>NLP</b>	1	7	0
<b>Image Processing</b>	0	0	6

Waktu komputasi yang dibutuhkan *Labeled Latent Dirichlet Allocation* menggunakan *unigram* + tanpa TF-IDF + *single label* adalah 1 menit 30 detik, sendangkan untuk penggunaan *bigram* + tanpa TF-IDF + *single label* adalah 4 menit 30 detik. Berdasarkan hipotesa yang telah ditulis pada bagian latar belakang, dengan menggunakan metode *feature selection* (TF-IDF) pada *Labeled Latent Dirichlet Allocation* tingkat akurasi yang dihasilkan akan meningkat serta waktu komputasi akan lebih cepat. Dalam pengujian ini, penggunaan *Labeled Latent Dirichlet Allocation* menggunakan TF-IDF memiliki tingkat akurasi terbaik ketika menggunakan 600 TF-IDF dengan waktu komputasi yaitu 27 detik. Walaupun hasil TF-IDF yang dihasilkan lebih kecil dari penggunaan tanpa TF-IDF, penggunaan TF-IDF dengan memilih 600 TF-IDF tertinggi dapat dikatakan baik juga karena tingkat akurasi yang dihasilkan adalah 90% dan waktu komputasi yang dibutuhkan hanya 27 detik.

## BAB V

### KESIMPULAN DAN SARAN

#### 5.1 Kesimpulan

Di bawah ini merupakan kesimpulan yang didapat berdasarkan penelitian kategorisasi dokumen dengan menggunakan *Labeled Latent Dirichlet Allocation* dengan TF-IDF.

1. Penggunaan *bag of words* untuk *bigram* dan *trigram* menggunakan n TF-IDF tertinggi kurang sesuai untuk digunakan pada metode *Labeled Latent Dirichlet Allocation*, karena berdasarkan pengujian hasil akurasi yang dihasilkan kurang baik (di bawah 40%).
2. Penggunaan metode *Labeled Latent Dirichlet Allocation* dengan kombinasi fitur TF-IDF, *unigram*, dan *single label* menghasilkan nilai rata-rata *F-Measure* sebesar 80% hingga 90%, sedangkan untuk kombinasi fitur TF-IDF, *unigram*, dan *multi label* menghasilkan nilai rata-rata *F-Measure* sebesar 50% hingga 60%.
3. Hasil akurasi menggunakan *single label* lebih baik dibandingkan menggunakan *multi label* karena pada kasus *single label*, label yang dihasilkan oleh sistem hanya berjumlah 1 sehingga ketika terdapat 1 label yang cocok, maka sistem akan menganggap hasil tersebut benar. Sedangkan untuk kasus *multi label*, label yang dihasilkan oleh sistem berjumlah lebih dari 1 sehingga jika terdapat 1 label saja yang tidak cocok, maka sistem akan menganggap hasilnya salah.
4. Penggunaan n TF-IDF terbaik terjadi ketika metode *Labeled Latent Dirichlet Allocation* menggunakan 600 TF-IDF tertinggi, jenis *bag of words* yang digunakan adalah *unigram*, dan keluaran dari sistem berupa *single label*. Hasil akurasi yang dihasilkan oleh kombinasi fitur tersebut adalah sebesar 90%.
5. Waktu pembelajaran tercepat untuk kasus *unigram*, *bigram*, dan *trigram* dengan *single label* terbaik terjadi ketika *unigram* menggunakan 600 TF-IDF tertinggi. Waktu yang dibutuhkan oleh kombinasi fitur tersebut adalah 27 detik.
6. Kasus terbaik untuk kategorisasi dokumen menggunakan *Labeled Latent Dirichlet Allocation* terjadi ketika tanpa menggunakan TF-IDF, jenis *bag of words* yang digunakan adalah *unigram* atau *bigram*, dan keluaran dari sistem berupa *single label*. Hasil akurasi yang dihasilkan oleh kombinasi fitur tersebut adalah sebesar 95%.
7. Penggunaan metode *Labeled Latent Dirichlet Allocation* tanpa TF-IDF lebih tinggi daripada menggunakan n TF-IDF karena ada satu kondisi dimana ketika menggunakan TF-IDF terdapat beberapa kata yang sebenarnya penting tetapi tidak terpilih. Hal ini dapat terjadi karena kata yang tidak terpilih tersebut lebih banyak kemunculannya di beberapa dokumen dibanding kata yang lainnya, sehingga nilai TF-IDF untuk kata tersebut akan bernilai kecil.

## 5.2 Saran

Di bawah ini merupakan saran yang didapatkan berdasarkan penelitian yang perlu dipertimbangkan untuk pengembangan penelitian ke depannya.

1. Memperbanyak data *training* yang digunakan dengan label yang bervariasi, karena dengan banyaknya data, kata-kata yang akan diolah dalam proses pembelajaran pun akan bertambah banyak dan akurasi menggunakan metode *Labeled Latent Dirichlet Allocation* juga akan meningkat.

## DAFTAR PUSTAKA

- [1] Harrington, P. (2012). Machine Learning in Action. Manning.
- [2] Tsai, S. C., Jiang, J. Y., dan Lee, S. J. (2010). A Mixture Approach for Multi-Label Document Classification. halaman 387 – 391. IEEE.
- [3] Feng, G. Z., Guo, J., Jing, B. Y., dan Sun, T. (2015). Feature Subset Selection Using Naive Bayes for Text Classification. halaman 1-8. ACM.
- [4] Bai, Y. dan Wang, J. (2015). News Classifications with Labeled LDA. jilid 1, halaman 75-83. IEEE.
- [5] Bird, S., Klein, E., dan Loper, E. (2009). Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit. O'Reilly Media, Inc.
- [6] Moschitti, A. (2003). Natural Language Processing and Automated Text Categorization. halaman 1-133. CiteSeerX.
- [7] Islam, S. M., Jubayer, F. E. M., dan Ahmed, S. I. (2017). A Support Vector Machine mixed with TF-IDF Algorithm to Categorize Bengali Document. halaman 191-196. IEEE.
- [8] Fitzgerald, M. (2012). Introducing Regular Expressions: Unraveling Regular Expressions, Step-by-Step. O'Reilly Media, Inc.
- [9] Jivani, M. A. G. (2011). A Comparative Study of Stemming Algorithms. halaman 1930-1938. IEEE.
- [10] Blei, D. M., Ng, A. Y., dan Jordan, M. I. (2003). Latent Dirichlet Allocation. jilid 3, halaman 993-1022. ACM.
- [11] Ramage, D., Hall, D., Nallapati, R., dan Manning, C. D. (2009). Labeled LDA: A supervised topic model for credit attribution in multi labeled-corpora. jilid 1, halaman 248-256. ACM.
- [12] Griffiths, T. L. dan Steyvers, M. (2004). Finding scientific topics. jilid 101, halaman 5228-5235. National Academy of Sciences.
- [13] Darling, W. M. (2011). A Theoretical and Practical Implementation Tutorial on Topic Modeling and Gibbs Sampling. halaman 1-10. Research Gate.

## LAMPIRAN