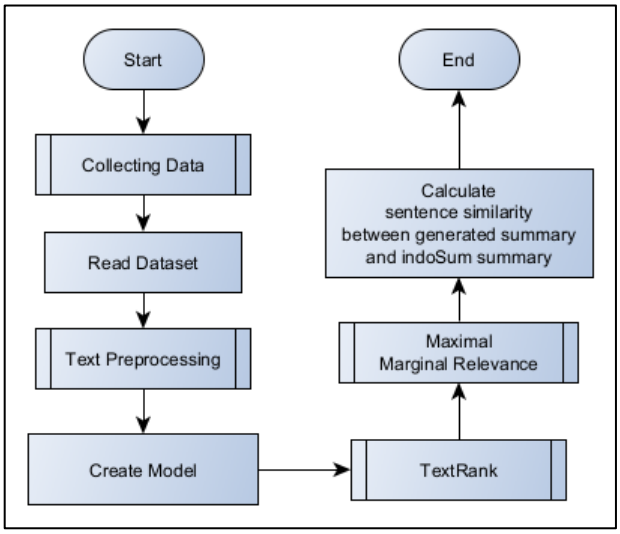
BAB 3

METODOLOGI PENELITIAN

3.1 Gambaran Umum Metodologi Penelitian

Gambaran umum metodologi penelitian berisi alur proses penelitian yang dilakukan dan spesifikasi sistem yang digunakan. Proses pertama yang dilakukan adalah mengumpulkan berita data difabel dalam *dataset* indosum. Kemudian, artikel berita akan di memasuki tahap *text preprocessing* agar artikel berita siap memasuki proses selanjutnya. Lalu aplikasi akan membuat dan melakukan *training model* dari artikel berita yang telah di-*preprocess*. Setelah itu, dilakukanlah proses perhitungan dan pemeringkatan menggunakan algoritma TextRank. Selanjutnya dilakukan perhitungan dan pemeringkatan ulang menggunakan metode Maximal Marginal Relevance. Lalu, menghitung nilai *cosine similarity* antara ringkasan hasil penelitian dengan ringkasan dari *dataset* IndoSum. *Flowchart* mengenai gambaran umum metodologi penelitian dapat dilihat pada Gambar 3.1.



Gambar 3.1 *Flowchart* Gambaran Umum Metodologi Penelitian

Spesifikasi *software* dan *hardware* yang digunakan dalam penelitian adalah sebagai berikut.

Software yang digunakan:

1) Python 3.7
2) Jupyter Notebook
3) Google Colaboratory
4) Flask
5) Ubuntu 20.04 LTS

Hardware yang digunakan:

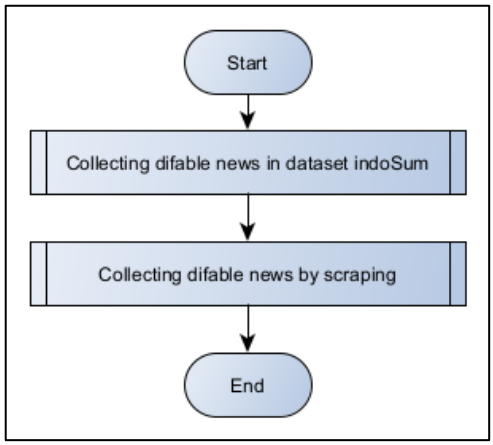
1) Processor: AMD Ryzen 5 2500u
2) Integrated Graphic Card: Radeon Vega 8 Graphics
3) Memory: 4 GB onboard DDR4 + 8 GB SODIMM

3.2 Studi Literatur

Studi literatur dilakukan dengan mencari, membaca dan memahami karya tulis ilmiah yang berhubungan dengan penelitian seperti Difabel, *Automatic Text Summarization*, *One-hot Encoding*, *Term Frequency – Inverse Document Frequency*, FastText, TextRank, Maximum Marginal Relevance, dan Teknik Evaluasi.

3.3 Collecting Data

Penelitian ini menggunakan dua buah *dataset* yaitu *dataset* IndoSum dan *dataset* hasil *scraping* peneliti terhadap beberapa *website* penyedia berita difabel. *Flowchart collecting data* pada kedua *dataset* dapat dilihat pada Gambar 3.2.



Gambar 3.2 *Flowchart Collecting Data*

Penjelasan mengenai setiap modul pada *flowchart collecting data* adalah sebagai berikut.

3.3.1 Collecting Difable News in Dataset IndoSum

IndoSum merupakan sebuah *dataset* yang berisi minimal 18.774 artikel berita *online*. Setiap artikel mengandung informasi mengenai judul, kategori, sumber berita, *URL* sumber artikel berita tersebut, dan ringkasan dengan

pendekatan abstraktif. Ringkasan tersebut dikerjakan secara manual oleh 2 orang penutur asli Bahasa Indonesia. Terdapat 6 kategori dalam *dataset* IndoSum, yaitu tajuk utama, showbiz, olahraga, hiburan, teknologi, dan inspirasi. Dataset IndoSum telah digunakan pada penelitian berjudul INDOSUM*: A New Benchmark Dataset for Indonesian Text Summarization* yang dilakukan oleh Kemal Kurniawan (2018). Gambaran singkat mengenai bentuk data pada *dataset* IndoSum dapat dilihat pada Gambar 3.3.

	category	gold_labels	id	paragraphs	source	source_url	summary
0	tajuk utama	[[False, True], [True, True], [False, False, F...	1501893029-lula-kamal-dokter-ryan-thamrin-saki...	[[[Jakarta, ,, CNN, Indonesia, -, -, Dokter, R...	cnn indonesia	https://www.cnnindonesia.com/hiburan/201708041...	[[Dokter, Lula, Kamal, yang, merupakan, selebr...
1	teknologi	[[False, False, False, False], [False, True, T...	1509072914-dua-smartphone-zenfone-baru-tawarka...	[[[Selfie, ialah, salah, satu, tema, terpanas,...	dailysocial.id	https://dailysocial.id/post/dua-smartphone-zen...	[[Asus, memperkenalkan, , ZenFone, generasi, ...
2	hiburan	[[True], [True], [False, False], [False], [Fal...	1510613677-songsong-visit-2020-bengkulu-perkua...	[[[Jakarta, ,, CNN, Indonesia, -, -, Dinas, Pa...	cnn indonesia	https://www.cnnindonesia.com/gaya-hidup/201711...	[[Dinas, Pariwisata, Provinsi, Bengkulu, kemba...
3	tajuk utama	[[True, True], [False, False, False], [True], ...	1502706803-icw-ada-kejanggalan-atas-tewasnya-s...	[[[Merdeka.com, -, Indonesia, Corruption, Watc...	merdeka	https://www.merdeka.com/peristiwa/icw-merasa-a...	[[Indonesia, Corruption, Watch, (, ICW,), mem...
4	tajuk utama	[[False, True], [True, True, True], [False], [...	1503039338-pembagian-sepeda-usai-upacara-penur...	[[[Merdeka.com, -, Presiden, Joko, Widodo, (, ...	merdeka	https://www.merdeka.com/peristiwa/usai-upacara...	[[Jokowi, memimpin, upacara, penurunan, bender...
...	...	...	...	...	...	...	...

Gambar 3.3 Bentuk *Dataset* IndoSum

Dataset IndoSum memiliki 5 buah bentuk kemudian masing-masing bentuk terbagi ke dalam 3 buah *subset* yaitu *training subset, test subset,* dan *development subset (Validation subset)*. Setiap bentuk mengandung 18774 artikel berita yang sama tetapi isi berita dalam setiap *subset* berbeda. Penelitian ini akan menggabungkan kembali setiap *subset* ke dalam sebuah *dataframe* sehingga perbedaan data pada setiap *subset* tidaklah berpengaruh. Struktur *file* pada *dataset* IndoSum dapat dilihat pada Gambar 3.4.

Gambar 3.4 Struktur *File* Pada *Dataset* IndoSum

Setiap bentuk memiliki proporsi yang hampir sama pada setiap subsetnya yaitu *training subset* (±75%), *test subset* (±20%) dan *development subset* (±5%). Jumlah data pada *training subset, test subset,* dan *development subset* dalam setiap bentuk dapat dilihat pada Gambar 3.5.

```
==========================================
DATASET - 1
length of dataset :  18774
length of subset train :  14262  || Ratio :  75.96676254394376
length of subset test  :  3762   || Ratio :  20.038350910834133
length of subset dev   :  750    || Ratio :  3.9948865452221156
==========================================
==========================================
DATASET - 2
length of dataset :  18774
length of subset train :  14263  || Ratio :  75.97208905933738
length of subset test  :  3762   || Ratio :  20.038350910834133
length of subset dev   :  749    || Ratio :  3.9895600298284863
==========================================
==========================================
DATASET - 3
length of dataset :  18774
length of subset train :  14290  || Ratio :  76.11590497496537
length of subset test  :  3737   || Ratio :  19.905188025993397
length of subset dev   :  747    || Ratio :  3.978906999041227
==========================================
==========================================
DATASET - 4
length of dataset :  18774
length of subset train :  14272  || Ratio :  76.02002769788004
length of subset test  :  3752   || Ratio :  19.98508575689784
length of subset dev   :  750    || Ratio :  3.9948865452221156
==========================================
==========================================
DATASET - 5
length of dataset :  18774
length of subset train :  14266  || Ratio :  75.98806860551827
length of subset test  :  3761   || Ratio :  20.033024395440503
length of subset dev   :  747    || Ratio :  3.978906999041227
==========================================
```

Gambar 3.5 Jumlah Data Pada *Training Subset, Test Subset*, dan *Development Subset* Dalam Setiap Bentuk

Dataset IndoSum bukanlah *dataset* eksklusif yang hanya berisi berita mengenai penyandang disabilitas. Sehingga dilakukanlah proses pemilihan berita difabel pada *dataset* IndoSum. Pemilihan berita dilakukan dengan memilih berita yang mengandung *keywords* berikut: 'difabel',' penyandang', ‘disabilitas’, ‘tuna’, 'tunarungu', 'rungu', 'tunanetra', 'netra', 'tunawicara', 'wicara', 'tunadaksa', 'daksa', 'tunalaras', ' tunagrahita' dan 'tunaganda' secara otomatis. Kemudian, artikel berita tersebut akan diperiksa secara manual untuk memastikan hanya berita yang membahas penyandang disabilitas yang digunakan dalam penelitian.

Pertama-tama aplikasi akan membaca *training subset, test subset,* dan *development subset* pada *dataset* IndoSum yang berbentuk jsonl kemudian ketiga *subset* tersebut digabungkan ke dalam sebuah *dataframe*. Aplikasi memeriksa setiap kata dalam artikel berita yang berada di dalam *dataframe.* Jika artikel berita mengandung salah satu kata dalam *keywords* maka aplikasi akan mencatat indeks artikel berita tersebut dan melanjutnya pencarian pada artikel berita selanjutnya.

Proses pemilihan berita yang mengandung keywords menghasilkan 67 berita terpilih. Kemudian dilanjutkan dengan proses pemeriksaan secara manual. Proses pemeriksaan berita secara manual dilakukan dengan membaca berita satu-persatu untuk memastikan hanya berita yang membahas penyandang disabilitas yang digunakan dalam penelitian. Proses pemeriksaan secara manual menghasilkan 51 artikel berita yang siap digunakan. Terdapat 16 artikel berita yang tidak relevan dan salah satu isi dari artikel berita tersebut dapat dilihat pada Gambar 3.6.

Flowchart proses *Collecting News About People with Disabilities in Dataset* IndoSum dapat dilihat pada Gambar 3.7.

>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

Index Berita - 2805.

JUARA.NET, BURIRAM - Tes pitlane menjadi salah satu sesi yang akan dijalani pebalap MotoGP pada tes pramusim di Sirkuit Chang Buriram, Thailand.

Pitlane adalah jalur yang menghubungkan lintasan utama dengan pit stop.

Hal itu dikarenakan adanya keluhan dari pebalap soal jalur pitlane yang berbahaya saat pertemuan komite keselamatan selepas hari pertama tes, Jumat (16 / 2 / 2018).

Para pebalap mengeluhkan potensi kecelakaan di jalur masuk dan jalur keluar pitlane.

(Baca Juga: Valentino Rossi Tidak Akan Menghitung Usianya untuk Perpanjang Kontrak bersama Tim Movistar Yamaha)

" Baik jalur masuk ke pit maupun jalur keluar adalah area yang akan bermasalah, jadi kami mencoba mencari cara untuk mengatasinya, sesuai dengan apa yang ada, " kata Race Director MotoGP, Mike Webb.

Pembenahan pada jalur keluar menjadi hal yang lebih dulu dilakukan karena seluruh area tersebut telah tertutup aspal yang juga menyatukannya dengan lintasan utama.

" Kami bisa sedikit mengarahkan pebalap sehingga mereka memiliki kesempatan lebih baik untuk melihat kondisi di dalam lintasan saat mereka keluar dari pit, " ujar Webb.

Sementara itu, pada jalur masuk ke pitlane akan memerlukan banyak modifikasi karena tidak menyatu dengan lintasan utama.

(Baca Juga: Asian Para Games 2018 - Tidak Mudah!

Ini Jalan Terjal Atlet Pelatnas Paracycling untuk Tampil di Asian Paragames)

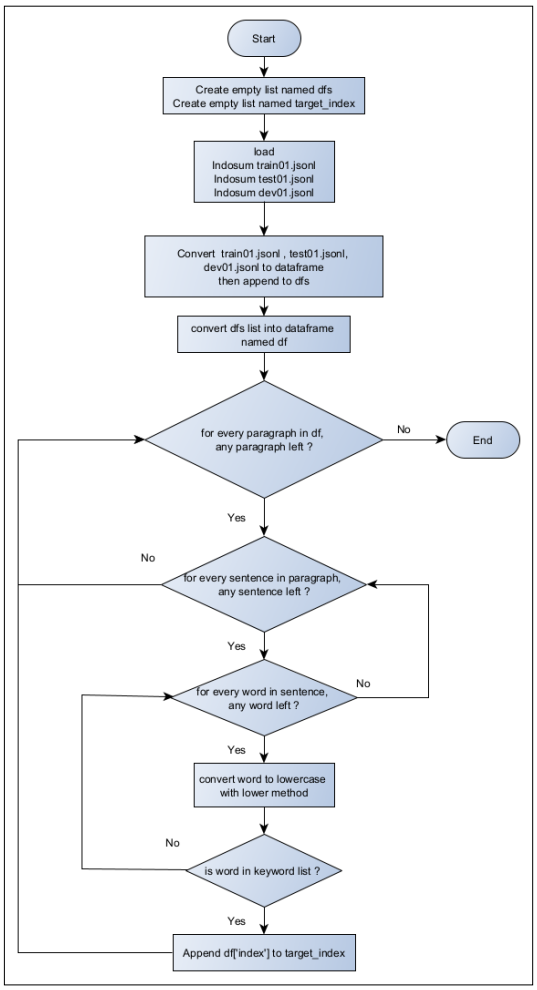
" Jadi kita punya beberapa hal untuk dicoba dan itulah gunanya uji coba "

" Kami akan berbicara dengan para pebalap di Qatar dan menanyakan bagaimana perbandingannya dengan jalur keluar yang asli, " ujarnya.

Tes pitlane tersebut akan digelar pada hari terakhir tes pramusim MotoGP di Thailand yang berlangsung pada Minggu (18 / 2 / 2018).

>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

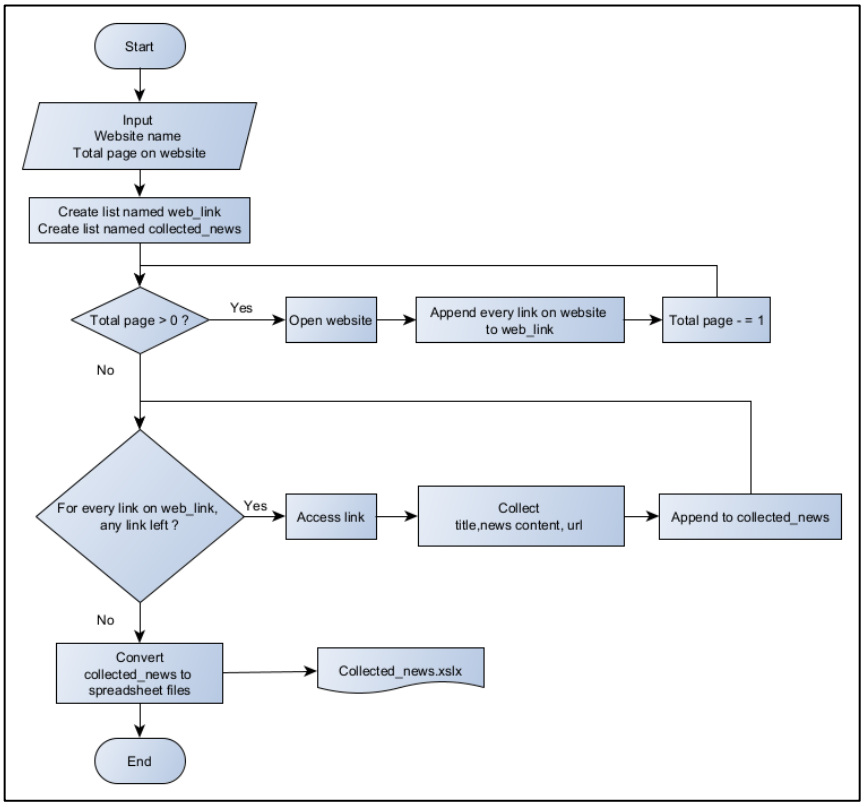
Gambar 3.6 Salah Satu Artikel Berita yang Tidak Relevan



Gambar 3.7 *Flowchart Collecting News About People with Disabilities in Dataset* IndoSum

3.3.2 Collecting Difable News by Scraping

Proses scapping pada *website* penyedia berita difabel dilakukan dengan menggunakan library *url*ib dan beautifulsoup pada Bahasa pemrograman Python. Library *url*ib membantu mengakses halaman *website* penyedia berita difabel sedangkan library beautifulsoup membantu membaca seluruh konten HTML pada halaman *website*. *Flowchart Collecting News about People With Disabilities by scraping* dapat dilihat pada Gambar 3.8.



Gambar 3.8 *Flowchart Collecting News about People with Disabilities by scraping*

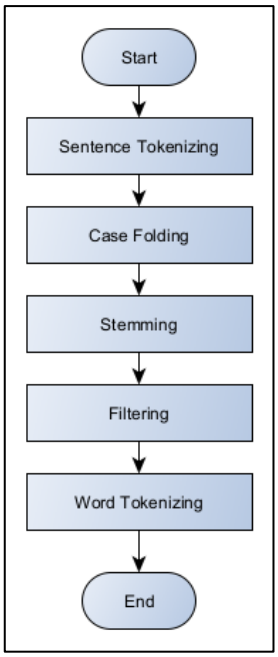
Langkah pertama saat melakukan *scraping* adalah dengan menentukan *website* yang mau di-*scraping*, penelitian ini melakukan *scraping* terhadap 4 (empat) buah *website* yaitu kompas, liputan6, newsdifabel dan tempo. Kemudian menentukan jumlah halaman *website* yang akan di-*scraping*. Setelah itu mengambil seluruh *link* pada setiap halaman *website* dan mengambil informasi seperti judul, *Uniform Resource Locator* (*url)* dan teks berita. Langkah terakhir adalah dengan menyimpan seluruh informasi hasil *scraping* ke dalam sebuah *file spreadsheet*. Proses ini menghasilkan 50 berita difabel yang kemudian dilakukan proses peringkasan manual oleh seorang pakar. Hasil proses scraping dan ringkasan pakar dapat dilihat pada Gambar 3.9.

Title	Text	URL	Summary
Kejujuran akan Disabilitas Membuat	seorang wanita yang mengesankan banya	https://www.liputan6.com/disabilitas/read/4453650/kejujuran-akan-disabilitas-membuc	text: Seorang wanita yang mengesankan banyak orang di The Bachelor dan menjadi kon pertama yang merupakan seorang tunarungu. Abigail lebih muda empat tahun (kini usia tahun) dari matt (29 tahun). Abigail lulus dari Linfield College pada 2017 dan bekerja di bidang keuangan. Saat ini bek sebagai analis keuangan klien di Opus Agency, di LinkedIn. Keterbukaan Abigail terkait disabilitasnya di sesi pertemuan pertama dengan cepat membuatnya menjadi favorit penggemar dan meningkatkan kesan pertama bagi Matt. Summary Correct: [1, 5]
Perjalanan Bima Kurniawan, Penyan	memiliki hambatan penglihatan atau tuna	https://www.liputan6.com/disabilitas/read/4506341/perjalanan-bima-kurniawan-penya	text: Memiliki hambatan penglihatan atau tunanetra tak menghentikan langkah Bima Ku
Manfaat Gabung Komunitas Anak Pe	jakarta memiliki anak dengan kelainan at	https://www.liputan6.com/disabilitas/read/4490062/manfaat-gabung-komunitas-anak-peny	Text: Angelman Syndrome adalah kelainan genetik di kromosom 15, di mana pada krom 15 ibu ada delesi atau hilangnya sebagian kromosom. Kelainan ini terbilang sangat langk memicu terjadinya disabilitas pada anak. Menurut Rani, hingga kini ia hanya bertemu 8 anak dengan sindrom yang sama di Indone Bahkan, komunitasnya pun belum ada. Guna mendapatkan dukungan dan berbagai pembelajaran, Rani memutuskan bergabung komunitas kelainan langka indonesia atau IRD. Dengan mengikuti komunitas, ia mengak sangat terbantu untuk mengetahui cara mengurus sang anak, Faustine Pitra Shabira yan disapa Utin. Summary correct: [4,5].
			Text: Penyanyi Yura Yunita memiliki pengalaman sendiri berkomunikasi dengan Tuli. Menurutnya, berbicara dengan teman Tuli ibarat bertemu dengan teman-te negara lain. Meskipun tak bisa bahasa asing, misalnya, teman saling mengerti d bahasa isyarat.

Gambar 3.9 *Dataset* Hasil Proses *Scraping*

3.4 Text Preprocessing

Text Preprocessing biasanya menjadi tahap pertama pada *Natural Language Processing (NLP) system* dan dapat meningkatkan performa aplikasi tersebut (Camacho-Collados and Pilehvar, 2017; Babanejad *et al.*, 2020). *Text Preprocessing* adalah proses merubah *unstructured text* menjadi *structured text* sebelum diproses lainnya dilakukan (Adelia, Suyanto and Wisesty, 2019). *Flowchart* pada proses *text preprocessing* dapat dilihat pada Gambar 3.10.



Gambar 3.10 *Flowchart text preprocessing*

Penjelasan mengenai setiap proses dalam *flowchart text preprocessing* adalah sebagai berikut.

1. Flowchart Sentence Tokenizing

Pertama-tama, artikel berita akan memasuki tahap *sentence tokenizing* yaitu pemecahan paragraf menjadi kalimat-kalimat. Proses *sentence tokenizing* ini menggunakan *method sent_tokenize* dari *library* NLTK. Hasil dari proses *sentence tokenizing* disimpan di dalam sebuah *list* bernama *sentences*.

2. *Case folding*

Paragraf yang telah dipecah menjadi kalimat akan memasuki proses *case folding. Case folding* atau *lowercasing* adalah Teknik *preprocessing* yang paling mudah dan bekerja dengan cara merubah semua huruf kapital (*uppercase*) dalam setiap kalimat menjadi huruf kecil (*lowercase*) (Gunawan *et al.*, 2016; Camacho-Collados and Pilehvar, 2017). Proses *case folding* ini menggunakan *method lower* pada bahasa pemrograman python. Hasil dari proses *case folding* ini akan disimpan kembali ke dalam *list sentence*.

3. *Stemming*

Pada proses *stemming*, setiap kata pada kalimat hasil *case folding* akan diubah menjadi bentuk kata akarnya (*root*) masing-masing. *Stemming* bekerja dengan cara menghapus imbuhan pada awalan dan akhiran kata sehingga setiap kata memiliki representasi yang sama (Sabuna and

Setyohadi, 2018; Babanejad *et al.*, 2020). Proses stemming ini menggunakan *stemmer* berbahasa indonesia dari *library* sastrawi. Hasil dari proses *stemming* ini disimpan ke dalam sebuah *list* dengan nama *stemmed*.

4. *Filtering*

Pada proses *filtering*, dilakukan pembuangan *stopword* dari kalimat hasil *stemming* (Ferré *et al.*, 2020). *Stopword* adalah kata-kata yang sangat sering muncul dalam suatu bahasa (Babanejad *et al.*, 2020). Proses pembuangan kata tidak penting (*stopword*) dilakukan dengan mengecek pada kamus *stopword*. Jika terdapat kata yang juga berada dalam kamus *stopword* maka kata tersebut akan dihapus dari daftar kata. Proses *filtering* ini menggunakan kamus *stopword* dari *library* Sastrawi. hasil dari proses *filtering* disimpan ke dalam sebuah *list* dengan nama *filtered*.

5. *Word Tokenizing*

Kalimat yang sudah melewati tahap *filtering* akan dipecah menjadi menjadi kata-kata berdasarkan spasi yang terdapat antara setiap kata pada tahap *word tokenizing / word segmentation* (Ahmadi, 2020; Seeha *et al.*, 2020). Proses word tokenizing ini menggunakan *method word_tokenizer* dari *library* NLTK.

3.5 Create Model

Terdapat 5 permodelan yang diuji adalah *one-hot encoding*, *term frequency – inverse document frequency* (TF-IDF), FastText *pre-trained* model, FastText CBOW (*continuous bag of words*) dan Fasttext Skipgram. Hanya satu metode

dengan nilai evaluasi terbaik diantara kelima model tersebut yang akan diimplementasikan pada aplikasi. Pada pengujian terhadap *one-hot encoding,* aplikasi akan menggabungkan kata-kata yang sudah melewati tahap *text preprocessing* sebuah *word dictionary* untuk memudahkan perhitungan *similarity* nantinya. Pada permodelan *term frequency – inverse document frequency* (TF-IDF), FastText CBOW (*continuous bag of words*) dan Fasttext Skipgram aplikasi akan menginisiasi model dan melatihnya menggunakan kata-kata hasil *text preprocessing*. Sedangkan pada permodelan FastText *pre-trained* model aplikasi hanya cukup melakukan *load* model. Penjelasan untuk setiap proses pembuatan model adalah sebagai berikut.

1. One-Hot Encoding

   Pembentukan *word dictionary* diawali dengan membaca list hasil *text preprocessing*. List hasil *text preprocessing* merupakan sebuah list 2 (dua) dimensi dimana dimensi pertama berisi kalimat pada artikel berita dan dimensi kedua berisi kata dalam kalimat tersebut. Aplikasi akan menggabungkan setiap item pada list hasil *text preprocessing* menjadi sebuah list 1 (dimensi) bernama model. Pada akhirnya aplikasi akan menghilangkan kata *redundant* pada *list* model. Perbedaan antara *word dictionary* yang dibentuk dengan *list* hasil *text preprocessing* dapat dilihat pada Gambar 3.11.

```
>>>>>>>>>>>
Preprocessed List =  [  ['jakarta', 'pos', 'kota', 'tingkat', 'layan', 'bus', 'transjakarta', 'apresiasi', 'warga'],
                        ['kecuali', 'warga', 'milik', 'butuh', 'khusus', 'sandang', 'disabilitas']]
>>>>>>>>>>>
Word Dictionary   =  [  'jakarta', 'pos', 'kota', 'tingkat', 'layan', 'bus', 'transjakarta', 'apresiasi', 'warga', 'kecuali',
                        'milik', 'butuh', 'khusus', 'sandang', 'disabilitas']
>>>>>>>>>>>
```

Gambar 3.11 Hasil *Word Dictionary*

2. Term Frequency – Inverse Document Frequency (TF-IDF)

Proses pembangunan dan pelatihan model TF-IDF diawali dengan membaca list hasil *text preprocessing.* Aplikasi akan membangun model TF-IDF menggunakan *class* TfidfVectorizer dari *library* scikit-learn. Kemudian aplikasi melakukan *train* pada model yang sudan dibangun menggunakan data hasil *preprocessing*.

3. FastText Pre-Trained Model

FastText menyediakan *pre-trained* model dengan arsitektur CBOW (*continuous bag of words*) dalam Bahasa Indonesia. *Pre-trained* model adalah sebuah model *machine learning* yang sudah dilatih dengan berbagai macam data sebelumnya dan siap digunakan oleh siapa saja tanpa perlu melakukan *training* ulang. *Pre-trained* model dari FastText sendiri sudah dilatih menggunakan Common Crawl dan Wikipedia. Penggunaan FastText *pre-trained* model dalam Bahasa Indonesia sendiri masih terbilang berguna karena model dapat merepresentasikan kumpulan kata baik secara *meaning-wise* dan *context-wise* (Young and Rusli, 2019). Pada tahap ini aplikasi hanya cukup melakukan *load* pada *pre-trained* FastText model yang bernama cc.id.300.bin.

4. FastText Skipgram Model

Proses pembangunan dan pelatihan model FastText dengan arsitektur skipgram diawali dengan membaca list hasil *text preprocessing.* Aplikasi akan membangun model FastText menggunakan *function* train_unsupervised dari *library* FastText dengan mengatur nilai parameter model dengan skipgram. Kemudian aplikasi melakukan *train* pada model yang sudan dibangun menggunakan data hasil *preprocessing*.

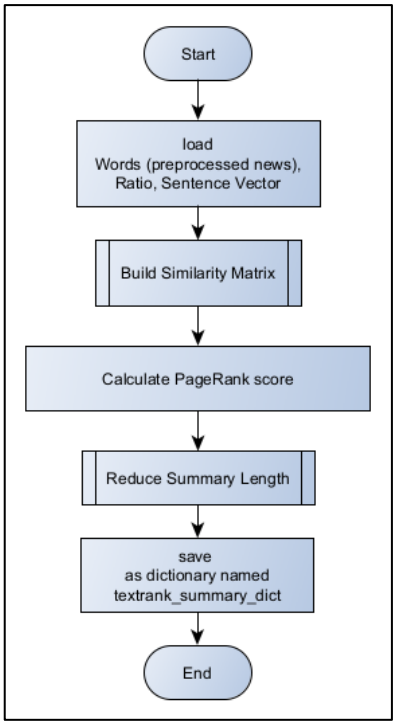
5. FastText CBOW Model

Proses pembangunan dan pelatihan model FastText dengan arsitektur skipgram diawali dengan membaca list hasil *text preprocessing.* Aplikasi akan membangun model FastText menggunakan *function* train_unsupervised dari *library* FastText dengan mengatur nilai parameter model dengan cbow. Kemudian aplikasi melakukan *train* pada model yang sudah dibangun menggunakan data hasil *preprocessing*.

3.6 TextRank

TextRank akan membentuk *similarity matrix* dengan memanfaatkan word dictionary / model yang sudah dibentuk sebelumnya. Kemudian *similarity matrix* tersebut akan representasikan ke dalam bentuk graf. Algoritma TextRank akan melakukan perhitungan *pagerank score* pada setiap graf yang sudah dibuat dan mengurutkan kalimat berdasarkan dengan *pagerank score*. Beberapa kalimat dengan *pagerank score* terendah akan dihapus / tidak diikutsertakan pada kalimat ringkasan. Hasil dari proses TextRank akan disimpan pada sebuah *dictionary*

dengan kalimat sebagai *keys* dan *pagerank score* sebagai *value* nya. *Flowchart* mengenai proses TextRank dapat dilihat pada Gambar 3.12.



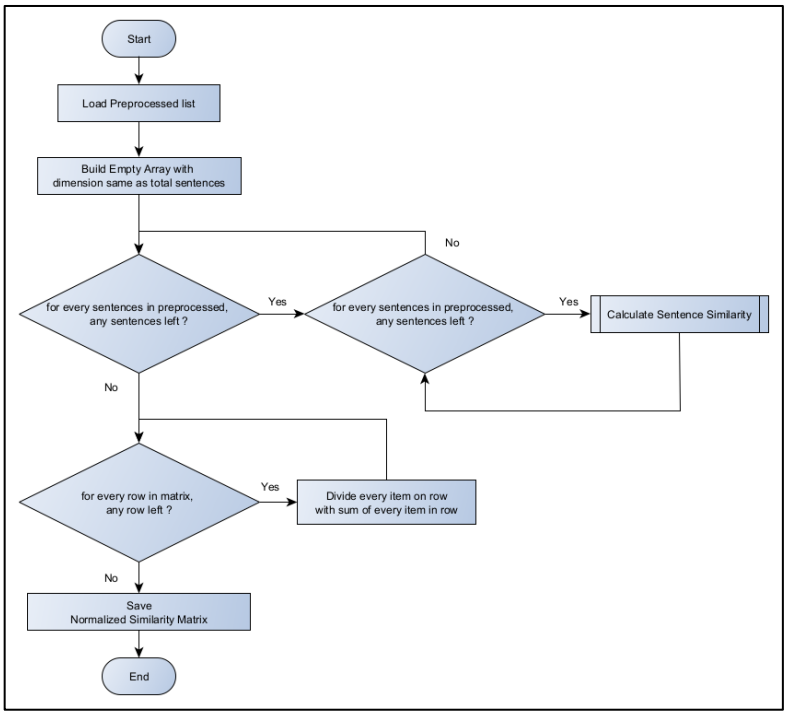
Gambar 3.12 *Flowchart* Proses TextRank

Penjelasan mengenai setiap modul dalam *flowchart text preprocessing* adalah sebagai berikut.

3.6.1 Build Similarity Matrix

Similarity Matrix merupakan matriks yang menyimpan *similarity* antar setiap kalimat dalam artikel berita. *Similarity Matrix* memiliki ukuran $n \times n$ dimana n adalah banyak kalimat dalam artikel berita. *Similarity Matrix* dibuat dengan cara menghitung nilai *cosine similarity* antar kalimat. *Similarity* antara kalimat 1 dan 2 akan menempati index [1,2] pada *similarity matrix*, similarity antara kalimat 2 dan 3 akan menempati index [2,3] pada *similarity matrix* dan begitu seterusnya.

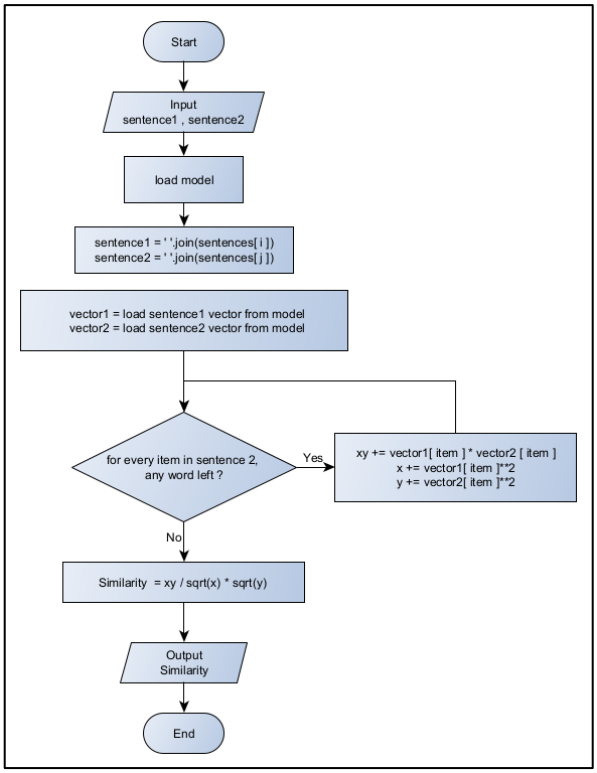
Sekarang nilai pada *similarity matrix* memiliki rentang nilai yang sangat besar maka dari itu diperlukan proses normalisasi. Penelitian ini melakukan normalisasi matrix secara *row-wise* dengan cara membagi setiap *item* pada baris ke-i dengan total nilai pada baris ke-i. *Flowchart* mengenai pembuatan *similarty matrix* dapat dilihat pada Gambar 3.13.



Gambar 3.13 *Flowchart Build Similarity Matrix*

Pada proses perhitungan *sentence similarity*, setiap kalimat akan direpresentasikan ke dalam bentuk *vector* dengan memanfaatkan *word dictionary* / model yang sudah dibuat. Kemudian dilakukan perhitungan *cosine similarity* terhadap vector tersebut. Nilai *similarity* pada penelitian ini dihitung menggunakan *cosine similarity*. Proses mendapatkan *vector* kalimat berbeda pada setiap

permodelan. *Flowchart calculate sentence similarity* dapat dilihat pada Gambar 3.14.



Gambar 3.14 *Flowchart Calculate Sentence Similarity*

Penjelasan mengenai proses mendapatkan *vector* kalimat pada tiap model adalah sebagai berikut.

1. One-hot Encoding

   Proses perhitungan *sentence similarity* pada model one-hot encoding dilakukan dengan menginisiasi 2 buah matriks / *vector* dengan dimensi sebesar *word dictionary*. Kedua *vector* diinisiasi dengan nilai 0 pada setiap index nya. *Vector* pertama merupakan representasi dari kalimat pertama dan vector kedua merupakan representasi dari kalimat kedua.

Ketika kata ke-i pada word dictionary ditemukan pada kalimat pertama maka nilai pada index ke-i vector pertama akan berubah menjadi 1. Ketika kata ke-i pada word dictionary ditemukan pada kalimat kedua maka nilai pada index ke-i vector kedua akan berubah menjadi 1.

2. Term Frequency – Inverse Document Frequency (TF-IDF)

   Proses perhitungan *sentence similarity* pada model TF-IDF dilakukan dengan merepresentasikan kalimat ke dalam bentuk *vector* dengan menggunakan method *transform* dari *library* Scikit-learn pada model yang sudah dibangun sebelumnya.

3. FastText

   Proses perhitungan *sentence similarity* baik pada model *pre-trained* FastText, FastText skipgram model, FastText CBOW model dilakukan dengan merepresentasikan kalimat ke dalam bentuk *vector* dengan menggunakan method *get_sentence_vector* dari *library* FastText.

3.6.2 Calculate PageRank Score

Langkah pertama dari proses ini adalah membaca terhadap *similarity matrix* yang sudah dibuat. *Similarity matrix* yang sudah dibuat akan direpresentasikan sebagai *graph*. *Graph* tersebut akan berisi *node* dan *edge* dimana setiap *node* merupakan representasi dari setiap kalimat dan *edge* merepresentasikan nilai *similarity* antar kalimat. Pembuatan graf dari *similarity matrix* ini menggunakan *method from_numpy_array* dari *library* networkX.

Proses perhitungan *pagerank score* sebuah kalimat dilakukan dengan mengaplikasikan rumus *pagerank* pada nilai kalimat dalam *similarity matrix*. Kemudian dilakukan perhitungan nilai *delta*, *delta* adalah selisih antara nilai yang lama dengan nilai yang baru. Perhitungan dilakukan secara terus-menerus hingga nilai *delta* menjadi lebih kecil daripada *eps (threshold)* dan berarti hasil perhitungan bersifat konvergen. *Threshold* (*eps*) pada algoritma *textrank* bernilai 0.0001 (Khan *et al.*, 2018; Ullah and Al Islam, 2019). Hasil dari perhitungan *pagerank score* diurutkan berdasarkan nilai *pagerank score* terbesar menggunakan *method sort*. Perhitungan *pagerank score* menggunakan *method pagerank_numpy* yang juga sudah disediakan oleh *library* networkX. Setelah mendapatkan *pagerank score* dari setiap kalimat, langkah selanjutnya adalah mengurangi jumlah kalimat dalam ringkasan dengan cara menghapus sebagian kalimat dengan nilai *pagerank score* terendah.

3.6.3 Reduce Summary Length

Langkah pertama dalam mereduksi jumlah kalimat adalah dengan membaca *variable* ranked_pagerank yang memiliki tipe data *dictionary*. Kemudian dibentuklah sebuah *array* dengan nama ranked_sentences yang berisi setiap key dalam ranked_pagerank dan *array* dengan nama ranked_score yang berisi setiap *value* dalam ranked_pagerank. Kemudian aplikasi akan melakukan proses *looping*, dengan setiap iterasinya akan menghapus indeks terakhir pada *array* ranked_score dan ranked_sentences. Proses *looping* akan berlangsung sebanyak limit yang ditentukan.

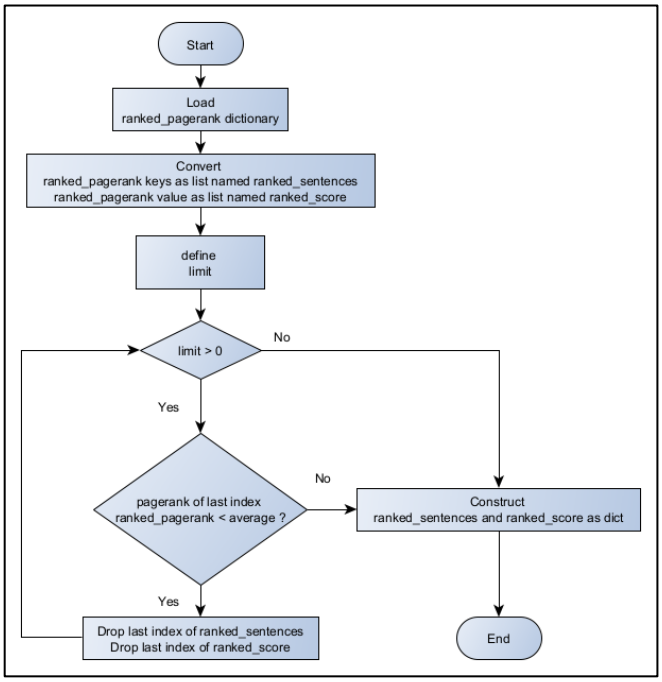
Terdapat 2 cara penentuan *limit* yang diuji dalam penelitian ini, cara pertama akan disebut dengan metode *above average*. Pada metode *above average*, *limit* dibuat sebanyak jumlah kalimat yang diinginkan pada ringkasan akhir tetapi proses *looping* akan berhenti saat indeks terakhir pada ranked_score bernilai lebih besar dari nilai rata-rata pada ranked_score. Sehingga, hanya kalimat dengan nilai pagerank *score* diatas rata-rata yang tersisa pada ringkasan

Cara kedua akan disebut dengan metode 50:50. Pada metode 50:50 aplikasi akan membuat jumlah kalimat yang akan direduksi oleh metode TextRank sama dengan jumlah kalimat yang akan direduksi oleh Maximal Marginal Relevance nantinya. Misal, artikel berita yang ingin diringkas memiliki 20 kalimat dan *user* menginginkan ringkasan akhir terdiri dari 4 kalimat. Aplikasi perlu melakukan reduksi sebanyak 16 kalimat sehingga metode TextRank dan metode Maximal Marginal Relevance akan mengurangi masing masing 8 kalimat.

Perhitungan penentuan limit pada metode 50:50 dapat dilihat pada persamaan berikut.

$$limit = (ratio * 2) * length\ of\ original\ news\ \div 100 \quad \ldots(3.1)$$

Flowchart mengenai *reduce summary length* dapat dilihat pada Gambar 3.15.

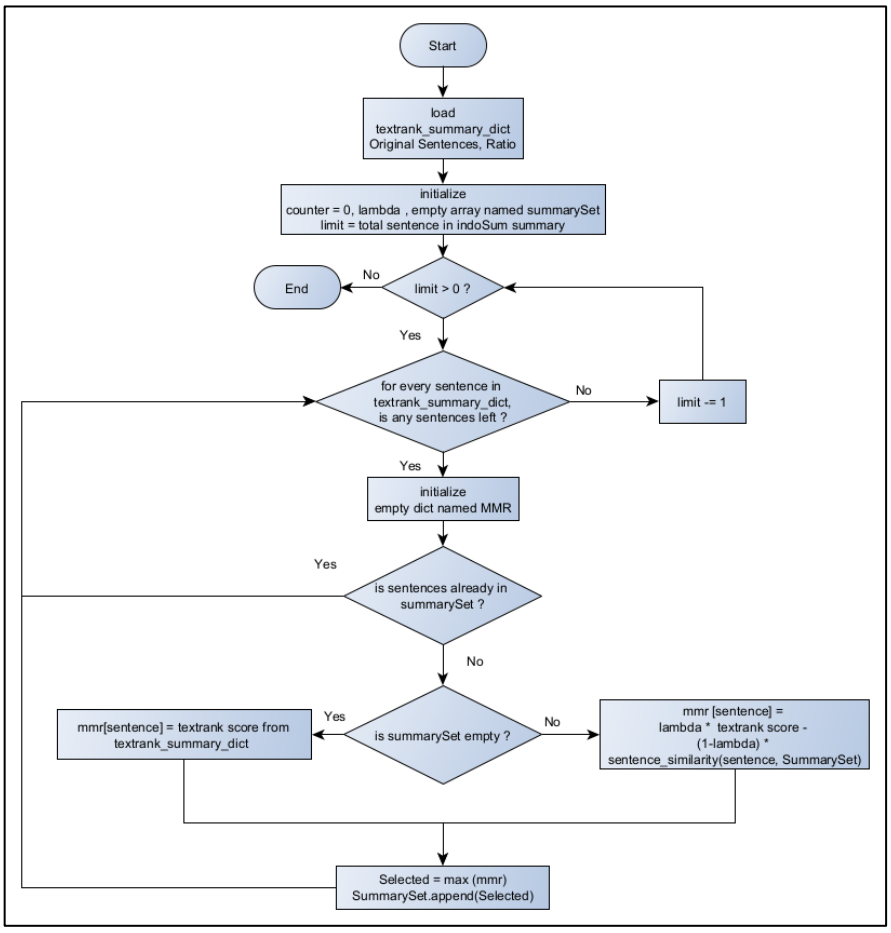


Gambar 3.15 *Flowchart Reduce Summary Length*

3.7 Maximal Marginal Relevance

Setelah artikel berita diringkas sebagian menggunakan metode TextRank, artikel berita akan diringkas kembali menggunakan metode Maximal Marginal Relevance. Sebelum memulai proses peringkasan, aplikasi akan membaca artikel berita yang asli, *ratio* hasil *input user*, kalimat yang terdapat pada hasil ringkasan Textrank dan juga nilai pagerank kalimat tersebut. Setelah itu, aplikasi akan menginisiasi nilai *lambda* (λ), *counter1* yang berguna untuk menentukan banyaknya kalimat yang akan diikutsertakan pada hasil akhir ringkasan, dan juga *array* kosong yang akan menampung kalimat-kalimat yang akan diikutsertakan pada ringkasan akhir dengan nama SummarySet. *Lambda* pada proses MMR merupakan parameter bobot yang dapat bernilai antara 0 hingga 1 dimana semakin rendah nilai *lambda* maka hasil tinggi *diversity* kalimat pada hasil ringkasan dan semakin tinggi nilai *lambda* maka semakin tinggi *relevancy* pada kalimat.

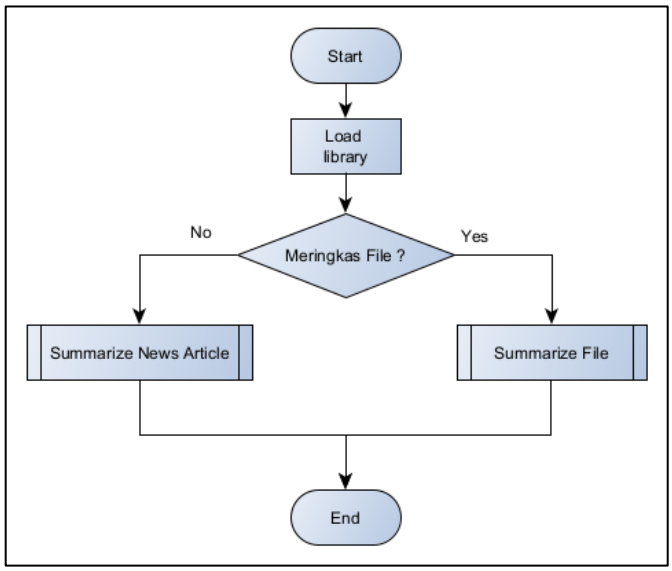
Aplikasi akan melakukan perhitungan MMR hingga jumlah kalimat akhir sudah sama dengan *counter1*. Pada iterasi pertama, aplikasi akan memberi bobot / *initial value* pada semua kalimat menggunakan nilai pagerank yang didapat dari proses sebelumnya dan menambahkan kalimat dengan bobot tertinggi ke dalam ringkasan akhir. Pada iterasi selanjutnya, aplikasi akan memberikan bobot pada setiap kalimat menggunakan perhitungan nilai MMR dan kalimat dengan nilai MMR tertinggi akan ditambahkan pada ringkasan akhir. *Flowchart* mengenai proses peringkasan menggunakan metode *Maximal Marginal Relevance* (MMR) dapat dilihat pada Gambar 3.16.



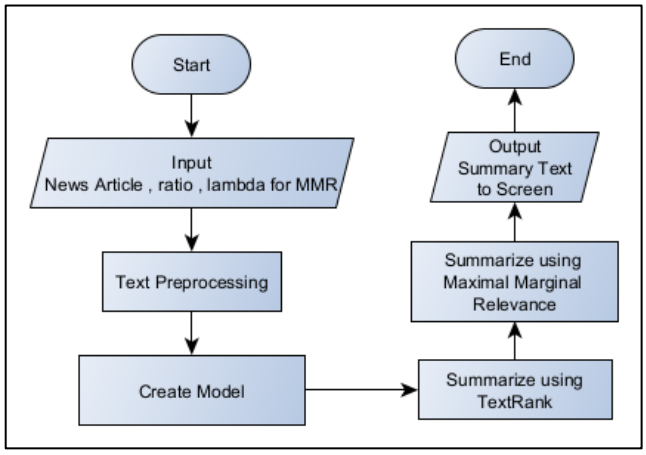
Gambar 3.16 *Flowchart Maximal Marginal Relevance*

3.8 Perancangan Aplikasi

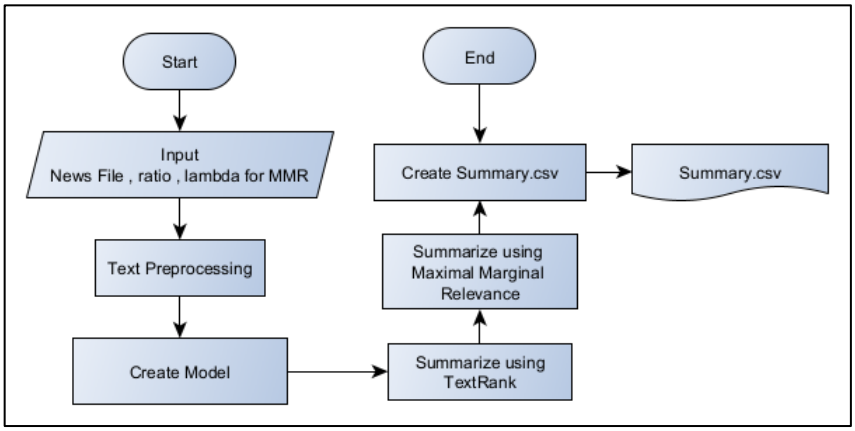
Perancangan aplikasi dilakukan untuk mengetahui alur kerja aplikasi yang dibuat dan proses pembuatan aplikasi tidak keluar dari alur yang telah ditentukan. Pemrograman aplikasi menggunakan *framework* Flask dan bahasa pemrograman Python. Aplikasi yang dibuat memiliki 2 fitur utama, yaitu meringkas berita teks dan meringkas file yang berisik kumpulan berita. Flowchart aplikasi peringkas berita otomatis dapat dilihat pada Gambar 3.17. *Flowchart* fitur meringkas teks berita dapat dilihat pada Gambar 3.18 dan *flowchart* fitur meringkas file dapat dilihat pada Gambar 3.19.



Gambar 3.17 *Flowchart* Aplikasi Peringkas Berita Otomatis



Gambar 3.18. *Flowchart* Fitur *Summarize News Article*



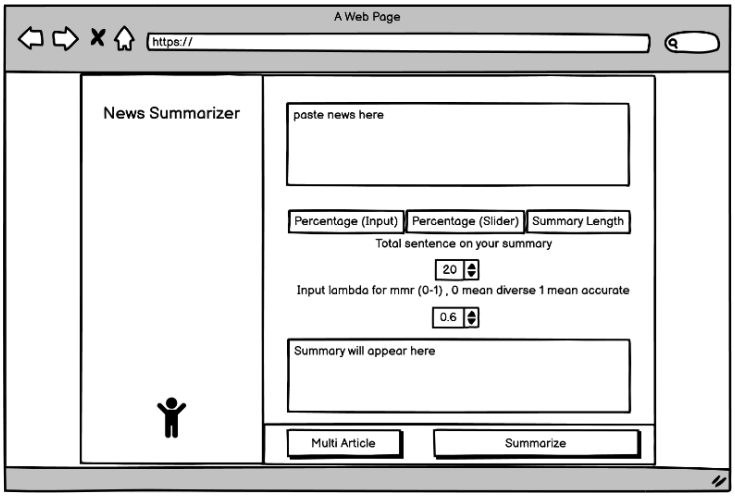
Gambar 3.19 *Flowchart* Fitur *Summarize File*

Pada fitur meringkas berita teks, proses diawali dengan *user* memasukkan teks berita yang ingin diringkas pada *textarea* yang sudah disediakan. Setelah itu, *user* perlu memilih rasio peringkasan guna menentukan seberapa panjang atau pendek hasil ringkasan yang akan dihasilkan oleh aplikasi.

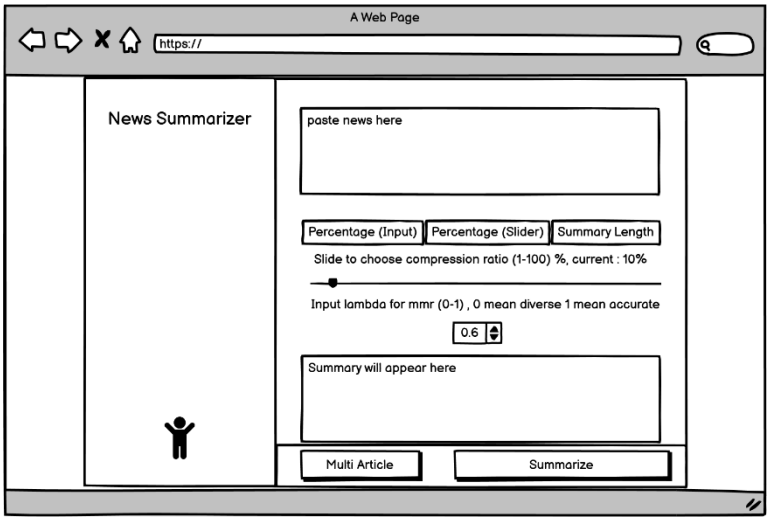
Terdapat 3 cara dalam menentukan panjang hasil ringkasan yaitu dengan melakukan input terhadap *percentage ratio*. *Percentage ratio* berkisar antara 1% hingga 100%. Jika *user* memilih peringkasan dengan *percentage ratio* 20% maka aplikasi akan menghasilkan teks ringkasan dengan panjang 20% dari panjang awal

teks berita sebelum diringkas. Kemudian *user* juga dapat memilih *percentage ratio* dengan menggunakan *slider*. Cara ketiga *user* juga dapat menentukan berapa banyak jumlah kalimat dalam hasil ringkasan akhir menggunakan *input box*.

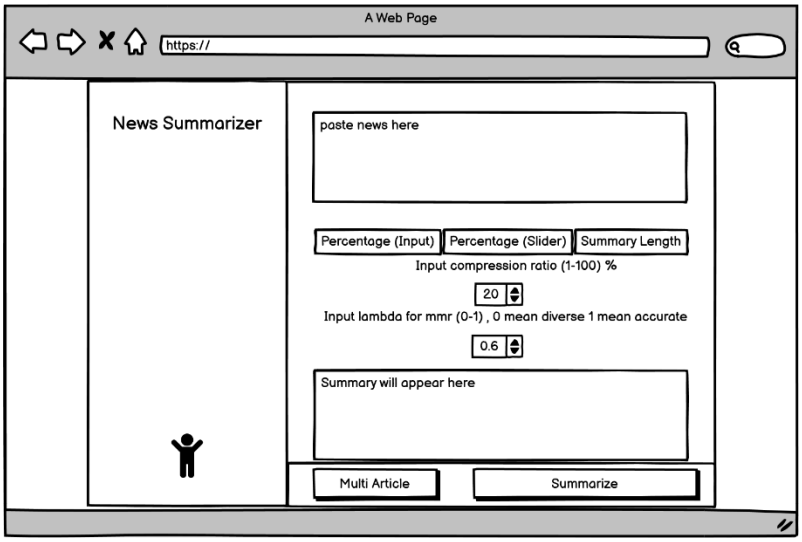
Setelah menentukan rasio peringkasan, *user* hanya perlu menekan tombol bertuliskan *Summarize*. Aplikasi akan melakukan *preprocessing* pada berita inputan user kemudian membuat model. Model yang dibuat tergantung pada model terbaik dari hasil pengujian. Jika model yang terpilih adalah FastText *pre-trained*, maka aplikasi cukup melakukan load model. jika bukan, maka aplikasi akan membagun model dan melakukan training model menggunakan data hasil preprocessing. Kemudian aplikasi akan meringkas menggunakan metode TextRank dan Maximal Marginal Relevance. Setelah itu, aplikasi akan menampilkan hasil ringkasan pada *textarea* yang sudah disediakan. Rancangan antarmuka meringkas teks berita dengan *compression* panjang kalimat dapat dilihat pada Gambar 3.20. Gambar 3.21 menunjukkan rancangan antarmuka meringkas teks berita dengan *compression percentage slider*. Rancangan antarmuka meringkas teks berita dengan *percentage input* dapat dilihat pada Gambar 3.22.



Gambar 3.20 Rancangan Antarmuka Meringkas Teks Berita dengan *Compression* Panjang Kalimat

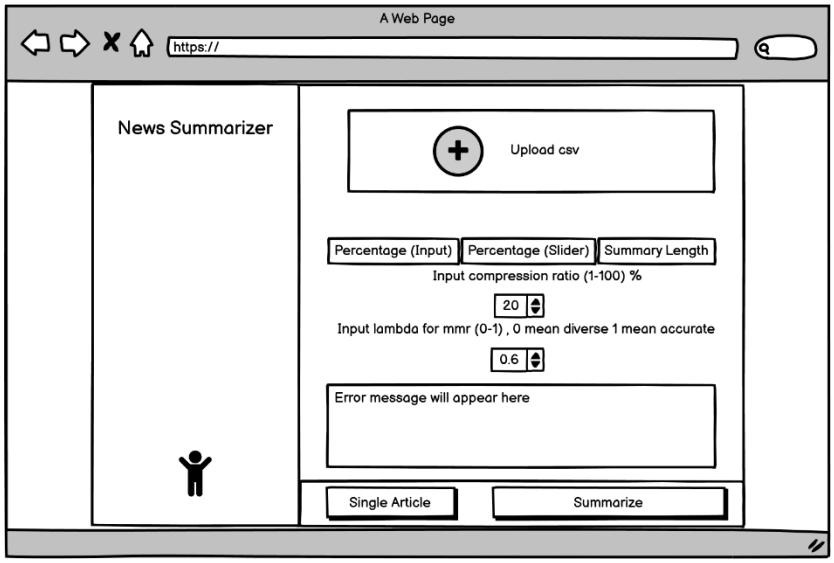


Gambar 3.21 Rancangan Antarmuka Halaman Meringkas Teks Berita dengan *Percentage Slider*



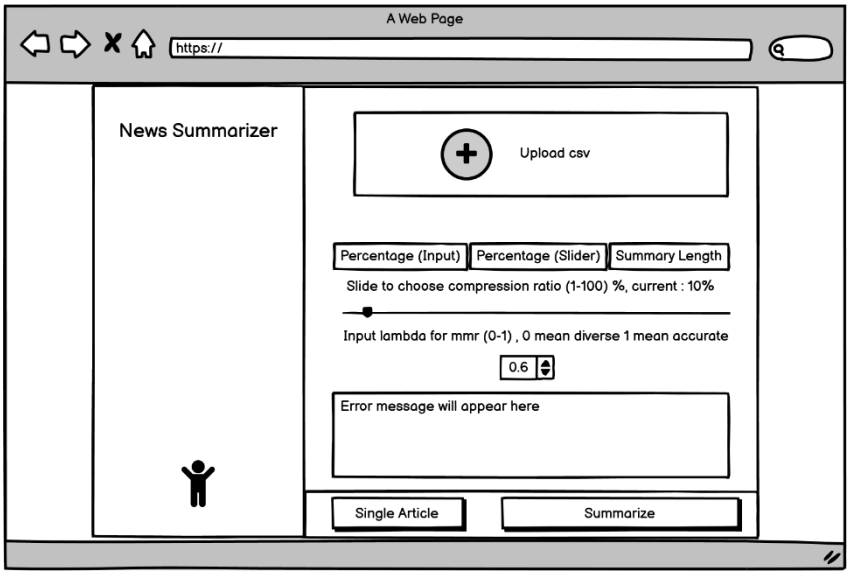
Gambar 3.22 Rancangan Antarmuka Meringkas Teks Berita Dengan *Percentage Input*

Fitur meringkas *file* dapat diakses dengan pada halaman *multi document*. *User* cukup melakukan melakukan klik pada tulisan *multi article* pada *navigation bar*. *User* perlu mengupload sebuah *file comma-separated value* (csv) yang berisi banyak artikel berita yang ingin diringkas kemudian memilih *compression ratio* yang diinginkan. Jika *user* sudah memasukkan artikel berita dan *compression ratio*, *user* cukup melakukan klik pada tombol dengan tulisan *summarize* dan aplikasi akan men-*generate* hasil ringkasan dan membuat *user* men-*download* sebuah file csv yang berisi artikel berita asli dan hasil ringkasan. Rancangan antarmuka pada halaman *multi document* dengan *percentage input* dapat dilihat pada Gambar 3.23.



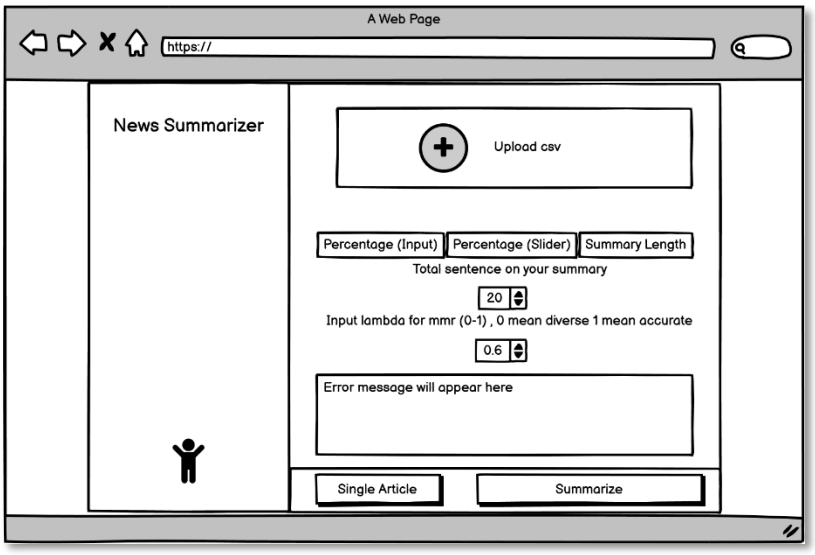
Gambar 3.23 Rancangan Antarmuka Halaman *Multi Document Percentage Input*

Rancangan antarmuka pada halaman *multi document* dengan *percentage slider* dapat dilihat pada Gambar 3.24.



Gambar 3.24 Rancangan Antarmuka Halaman *Multi Document Percentage Slider*

Rancangan antarmuka pada halaman *multi document* dengan *summary length* dapat dilihat pada Gambar 3.25.



Gambar 3.25 Rancangan Antarmuka Halaman *Multi Document Summary Length*

3.9 Pembangunan Aplikasi

Pembangunan aplikasi dilakukan dengan pembuatan aplikasi berbasis *web* dimana pengguna dapat meringkas artikel berita dan *file* yang berisi kumpulan berita. Saat meringkas artikel berita pengguna cukup memasukkan berita yang ingin diringkas pada *textbox* yang sudah disediakan dan aplikasi akan menampilkan hasil ringkasan dari berita yang dimasukkan oleh pengguna. Saat meringkas file kumpulan berita pengguna cukup mengunggah *file* dan aplikasi akan membuat peramban pengguna mengunduh *file* hasil ringkasan secara otomatis. Aplikasi akan dibuat menggunakan bahasa pemrograman python dengan *framework* flask.

3.10 Dokumentasi

Seluruh kegiatan mulai dari tahap studi literatur hingga kesimpulan akhir dari penelitian akan didokumentasikan dalam bentuk laporan. Agar proses dokumentasi dapat berjalan dengan baik dilakukanlah konsultasi dengan dosen pembimbing.