**BAB I**

**PENDAHULUAN**

* 1. **Latar Belakang Masalah**

Penyebaran berita *hoax* saat ini sedang marak terjadi, bahkan sudah masuk fase yang cukup mengkhawatirkan. Menteri Komunikasi dan Informatika (Menkominfo), Rudiantara mengatakan tidak kurang dari 900 ribu situs yang melakukan penyebaran berita *hoax* (Fajri, 2019). Penyebaran berita *hoax* ini bertujuan untuk menggiring opini para pembaca sehingga memiliki persepsi yang sama dengan pelaku. Berdasarkan pengamatan terhadap berita hoax yang beredar, ternyata berita hoax memiliki pola-pola tertentu dalam penulisan beritanya yang dilakukan dan disebarkan secara masif (Assidik, 2018). Beberapa contoh dari pola berita *hoax* seperti pesan berantai yang terdapat anjuran untuk menyebarkan, penggunaan tata bahasa yang kurang baik, tidak logis dan tidak konsisten serta bertentangan dengan akal sehat (Nasution, 2017). Apabila penyebaran berita *hoax* dibiarkan akan berdampak buruk bagi masyarakat seperti pengaburan fakta, hilangnya kepercayaan masyarakat hingga dapat mengakibatkan perpecahan antar masyarakat.

Ada beberapa media yang sering digunakan untuk melakukan penyebaran berita *hoax* seperti media cetak, email, situs web dan media sosial. Dari beberapa media penyebaran *hoax* tersebut, menurut hasil survei Masyarakat Telematika Indonesia (Mastel) menunjukkan bahwa media sosial menjadi sumber utama peredaran berita *hoax* (Librianty, 2017). Hal tersebut karena tingginya pengguna media sosial di Indonesia yang mencapai 150 juta orang atau sebesar 56% dari total populasi. Dari data tersebut juga menujukkan bahwa pengguna media sosial meningkat 15% dari tahun sebelumnya (Wearesocial, 2019). Ada beberapa jenis berita *hoax* yang sering diterima masyarakat seperti isu di bidang sosial politik, SARA, kesehatan, IPTEK, bencana alam, dan lain-lain. Banyaknya berita *hoax* yang tersebar membuat sulitnya masyarakat dalam memilah dan membedakan mana berita *hoax* dan tidak. Ada beberapa masyarakat yang kritis terhadap berita yang tersebar, lalu melakukan pengecekan apakah berita tersebut *hoax* atau tidak. Namun, tidak sedikit pula masyarakat yang langsung percaya terhadap berita yang disampaikan. Maka dari itu, perlu dikembangkan sebuah sistem yang dapat digunakan untuk mengklasifikasikan suatu konten berita sehingga dapat diketahui apakah berita tersebut *hoax* atau bukan.

Penelitian yang berkaitan dengan *hoax* pernah dilakukan menggunakan metode *naïve bayes* dengan tingkat akurasi 80% untuk mengklasifikasikan 3 kategori yaitu berita palsu (*hoax*), berita asli (*fact*) dan berita yang tidak berindikasi (Tanjung, 2018). Selain itu, ada penelitian lain yang membandingkan antara algoritma *rocchio* dengan *multinomial naïve bayes* dengan akurasi yang didapatkan sebesar 83,501% untuk *rocchio* dan 65,835% untuk *multinomial naïve bayes* (Afriza & Adisantoso, 2018). Algoritma *support vector machine* pun pernah digunakan untuk klasifikasi artikel *hoax* dengan tingkat akurasi yang didapatkan sebesar 95,8333% (Maulina & Sagara, 2018). Ada beberapa metode lain yang pernah digunakan seperti *gradient descent backpropagation* dengan tingkat akurasi 72,04% sedangan *lavenberg-marquad backpropagation* memiliki akurasi 72,19% dalam klasifikasi pengguna media sosial (Lhaksmana, Nhita, & Budhiarto, 2017). Penerapan *deep learning* juga pernah dilakukan dalam klasifikasi sentimen analisis *tweet* dengan menggunakan metode *deep belief network* dengan tingkat akurasi sebesar 93,31% (Zulfa & Winarko, 2017).

Ada tantangan yang harus dihadapi dalam proses klasifikasi berita *hoax* yaitu banyaknya berita *hoax* yang memiliki tingkat kemiripan yang sangat tinggi dengan berita aslinya. Bahkan dalam beberapa kasus, penambahan atau perubahan beberapa kata dari berita asli akan merubah maksud dan tujuan dari berita tersebut. Seperti kendala dari penggunaan metode *naïve bayes*, dimana kemiripan data membuat sulitnya melakukan klasifikasi berita *hoax* sehingga mempengaruhi tingkat error dan akurasinya (Tanjung, 2018). Selain itu, banyaknya data yang digunakan membuat waktu yang diperlukan untuk melakukan *preprocessing* dan *training* semakin lama. Metode-metode yang telah digunakan sebelumnya tidak dapat menemukan pola berita *hoax*, jadi hasil klasifikasinya tidak berdasarkan pola melainkan berdasarkan kemiripan data . Untuk meminimalisisr hal tersebut, dapat dilakukan dengan menggunakan metode *deep learning*.

Kelebihan dari penggunaan metode *deep learning* tersebut karena memiliki algoritma dengan struktur dan jumlah jaringan syaraf yang sangat banyak sebagai jaringan tersembunyi (*hidden layer*) yang dapat digunakan untuk berbagai hal, salah satunya klasifikasi teks. Metode *deep learning* yang dapat digunakan untuk klasifikasi teks yaitu *Convolutional Neural Network (CNN), Recurrent Neural Network (RNN),* dan *Hierarchical Attention Network (HAN)*. Ketiga metode tersebutmemiliki layer hirarki banyak/bertumpuk-tumpuk sehingga kendala yang berkaitan dengan kemiripan antar data dapat diminimalisir. Dengan penggunaan metode-metode tersebut, diharapkan juga dapat menemukan pola dari berita *haox* sehingga hasil yang didapat pun menjadi lebih akurat. Kelebihan lain yang dimiliki metode-metode tersebut yaitu menghilangkan ekstraksi fitur secara manual, tidak memerlukan informasi terhadap data yang akan dipelajarinya, dan dapat dikombinasikan dengan *Graphics Processing Unit* (GPU) untuk mempercepat proses *preprocessing* maupun *training*. Algoritma dari *deep learning* dapat melakukan *tuning* (penyetelan) dan pemilihan model yang paling optimal secara mandiri. Selain itu, metode tersebut terbukti memiliki tingkat akurasi yang tinggi dan efisien dalam proses pengklasifikasian. Seperti penelitian yang pernah dilakukan oleh Razi (2017) tentang Klasifikasi Artikel Berita Berbahasa Indonesia Menggunakan *Convolutional Neural Network* dengan nilai akurasi sebesar 96,70% dan presisi, recall serta f-measure mencapai 96,60%. Ketiga metode tersebut juga pernah dibandingkan untuk klasifikasi teks dengan hasil dari CNN, RNN, dan HAN secara berturut turut yaitu 96,12%, 95,76%, 95,6% menggunakan dataset pertama, 87,88%, 82,63%, 87,91% menggunakan dataset kedua, dan dengan dataset ketiga menghasilkan 91,6%, 89,54%, 92,16% (Maheshwari, 2018).

Oleh karena itu, penelitian ini akan mengkomparasikan metode *convolutional neural network, recurrent neural network* dan *hierarchical attention network* untuk klasifikasi berita *hoax.* Penelitian ini akan menggunakan data yang bersumber dari turnbackhoax.id yang telah memvalidasi berita *hoax* yang beredar di media sosial. Data tersebut akan menjadi data latih dan juga data uji. Penelitian ini diharapkan dapat mengetahui metode *deep learning* mana yang memiliki tingkat performa yang lebih baik. Selain itu, penelitian ini juga diharapkan dapat menghasilkan sebuah sistem yang dapat dimanfaatkan untuk melakukan klasifikasi berita berdasarkan pola-pola tertentu sehingga dapat diketahui apakah berita tersebut termasuk kategori hoax atau bukan.

* 1. **Perumusan Masalah**

Berdasarkan latar belakang yang telah dipaparkan, maka dapat diambil rumusan masalah sebagai berikut:

1. Bagaimana tingkat akurasi algoritma *convolutional neural network, recurrent neural network* dan *hierarchical attention network* dalam klasifikasi berita *hoax*?
2. Bagaimana tingkat efisiensi algoritma *convolutional neural network, recurrent neural network* dan *hierarchical attention network* dalam klasifikasi berita *hoax*?
   1. **Batasan Masalah**

Menghindari terjadinya topik permasalahan yang lebih luas, maka diberikan batasan masalah sebagai berikut:

1. Data penelitian didapatkan dari situs web turnbachoax.id yang telah memvalidasi berita *hoax* yang beredar di media sosial.
2. Data penelitian yang digunakan dari tahun 2015 sampai dengan 2019.
3. Data penelitian yang digunakan akan dikategorikan menjadi dua kelas yaitu *hoax* dan fakta.
4. Data penelitian yang digunakan berbahasa Indonesia.
5. Metode klasifikasi yang digunakan adalah metode *Convolutional Neural Network* (*CNN*)*, Recurrent Neural Network* (*RNN*)dan *Hierarchical Attention Network* (*HAN*).
6. Penelitian ini akan melakukan klasifikasi berdasarkan bentuk atau pola dari berita.
   1. **Tujuan Penelitian**

Tujuan pada penelitian ini adalah sebagai berikut:

1. Dapat mengetahui tingkat akurasi algoritma *convolutional neural network, recurrent neural network* dan *hierarchical attention network* untuk klasifikasi berita *hoax*.
2. Dapat mengetahui tingkat efisiensi algoritma *convolutional neural network, recurrent neural network* dan *hierarchical attention network* untuk klasifikasi berita *hoax*.
   1. **Manfaat Penelitian**

Dengan dilakukannya penelitian ini dapat mengetahui algoritma *deep learning* mana yang memiliki tingkat performa yang lebih tinggidalam menyelesaikan permasalahan tentang klasifikasi berita *hoax*. Selain itu, diharapkan dengan adanya sistem ini dapat memudahkan masyarakat khususnya pengguna media sosial dalam memilah dan menyaring berita yang tersebar. Sehingga dampak dari penyebaran berita *hoax* dapat diminimalisir.

* 1. **Metodologi Penelitian**

Metodologi penelitian yang digunakan pada penelitian ini adalah sebagai berikut:

* + 1. **Metodologi Pengumpulan Data**

Alur metodologi pengumpulan data yang digunakan pada penelitian ini adalah sebagai berikut:

1. Studi Literatur

Studi literatur dilakukan dengan cara mencari dan menghimpun data ataupun sumber-sumber pustaka yang dapat mendukung penelitian serta memberikan informasi untuk menyelesaikan permasalahan pada penelitian ini. Studi kepustakaan yang digunakan bersumber dari buku, jurnal, artikel dan paper yang berkaitan pada penelitian ini.

1. Pengumpulan Data

Data yang digunakan pada penelitian ini bersumber dari situs turnbackhoax.id. Situs tersebut telah memvalidasi kebenaran dari berita-berita yang beredar di media sosial. Pengumpulan data dilakukan dengan menggunakan teknik *web srcaping* dengan ketentuan data dari tahun 2015 sampai dengan data tahun 2019.

* + 1. **Metodologi Pengembangan Sistem**

Metodologi pengembangan sistem yang digunakan adalah metode *prototype.* Penggunaan metode *prototype* dikarenakan metode ini lebih menekankan pada komunikasi antara pengembang dan pelanggan sehingga kebutuhan pengguna dapat diterjemahkan dalam bentuk model (prototipe) dan hasil yang diharapkan pun dapat tercapai. Ada beberapa tahapan yang dilakukan pada metode *prototype* yaitu (Pressman, 2010):

1. *Communication*

Pada tahapan ini dilakukan proses komunikasi antara pengembang dan pelanggan mengenai tujuan dibuatnya suatu perangkat lunak. Selain itu, akan dilakukan analisis serta identifikasi kebutuhan apa saja yang akan diperlukan selama proses pembuatan perangkat lunak.

1. *Quick Plan dan Modeling Quick Design*

Pada bagian ini akan membuat perencanaan dan pemodelan secara cepat berdasarkan kebutuhan yang telah diidentifikasi pada tahap sebelumnya. Perencanaan yang akan dilakukan dapat berupa pembagian tugas dan rencana kerja. Sedangkan dari sisi pemodelan akan membuat desain model yang merepresentasikan aspek-aspek yang diinginkan dari pelanggan.

1. *Construction of Prototype*

Pada proses *contruction of prototype,* pengembang akan mulai membuat program (perangkat lunak) berdasarkan rencana dan model yang telah dibuat sebelumnya. Pada tahap ini juga termasuk *testing* dari perangkat lunak yang telah berhasil dibuat.

1. *Deployment Delivery & Feedback*

Setelah perangkat lunak berhasil dibuat dan telah melalui *testing,* proses selanjutnya yaitu tahap *deployment delivery & feedback*. Perangkat lunak tadi akan disampaikan kepada pelanggan untuk dilakukan pengecekan. Jika mendapatkan *feedback* baik, maka perangkat lunak akan diterima oleh pelanggan dan pembuatan perangkat lunak telah selesai. Apabila *feedback* yang diberikan kurang baik atau tidak sesuai dengan pelanggan, maka kembali ke tahap *communication* dengan membahas perbaikan yang perlu dilakukan oleh pengembang.

* 1. **Sistematika Penulisan**

Sistematika penulisan yang digunakan dalam menyusun laporan penelitian ini adalah sebagai berikut :

Bab I Pendahuluan

Pada bagian ini membahas tentang latar belakang masalah, perumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi penelitian, dan sistematika penulisan.

Bab II Tinjauan Pustaka

Tinjauan pustaka memuat tentang dasar teori yang digunakan untuk analisis dan perancangan sistem serta implementasi pada penelitian ini. Selain itu juga sebagai bahan referensi dan pondasi untuk memperkuat argumentasi dalam penelitian ini. Teori-teori yang sesuai dengan penelitian ini antara lain media sosial, *hoax,* klasifikasi, *deep learning,* *convolutional neural network, recurrent neural network,* dan *hierarchical attention network* .

Bab III Analisis dan Perancangan Sistem

Pada bagian ini akan membahas mengenai analisa dan perancangan sistem hingga implementasi sistem dalam mengklasifikasi berita *hoax* di media sosial.

Bab IV Hasil, Pengujian dan Pembahasan

Pada bab ini akan menyajikan hasil penelitian yang berisi hasil implementasi dari perancangan yang telah dibuat pada bab sebelumnya. Selain itu berisi pengujian terhadap hasil penelitian beserta pembahasannya.

Bab V Kesimpulan dan Saran

Pada bagian ini berisi kesimpulan dari hasil penelitian dan saran yang diajukan oleh penulis untuk pengembangan penelitian selanjutnya.

**BAB II**

**TINJAUAN PUSTAKA**

* 1. ***Hoax***

*Hoax* merupakan suatu kepalsuan yang disamarkan seolah-olah itu adalah kebenaran (MacDougall, 1958). *Hoax* dapat diartikan juga sebagai rangkaian informasi yang memang sengaja disesatkan, namun ‘dijual’ sebagai kebenaran (MacDougall, 1958). Beredarnya *hoax* dapat membentuk opini public yang mengarah kepada terjadinya kehebohan di masyarakat (MacDougall, 1958). Ada beberapa jenis informasi yang termasuk *hoax*, antara lain (Vibriza, Rahadi, Marwan, & Ahyad, 2017):

1. *Fake News* (berita palsu)
2. *Clickbait* (tautan jebakan)
3. *Confirmation Bias* (bias informasi)
4. *Misinformation* (informasi yang salah)
5. *Satire* (sindiran)
6. *Post-truth* (pasca kebenaran)
7. *Propaganda* (provokasi)

Dengan mudahnya penyebaran informasi pada saat ini, sangat memudahkan pula berita *hoax* beredar di masyarakat. Sayangnya, masih banyak masyarakat yang percaya dengan berita *hoax* yang beredar. Ada dua faktor yang menyebabkan masyarakat percaya terhadap berita *hoax*. *Pertama,* karena *hoax* yang dilakukan secara berulang-ulang dan dilakukan secara masif dapat dianggap menjadi suatu kebenaran. Selain itu, seseorang akan cenderung percaya terhadap berita *hoax* jika informasinya sesuai dengan opini atau sikap yang dimiliki (Respati, 2017).

Munculnya *hoax* di masyarakat dilatarbelakangi oleh berbagai tujuan, mulai dari main-main, pembentukan opini publik, penipuan hingga hasutan untuk memecah belah masyarakat. Berbagai tujuan tersebut memunculkan dampak negatif dari penyebaran berita *hoax*. Dampak negatif dari berita *hoax* tersebut seperti pengaburan fakta, hilangnya kepercayaan masyarakat hingga dapat mengakibatkan perpecahan antar masyarakat. Hal tersebut akan menjadi semakin buruk jika penyebaran berita *hoax* terus menerus dibiarkan.

Ada berbagai cara yang dapat dilakukan untuk terhindar dari berita *hoax*, seperti membaca seluruh isi berita, menelusuri sumber berita dan membandingkan antar berita. Selain itu, berita *hoax* juga biasanya memiliki pola-pola tertentu. Ada tujuh pola atau ciri-ciri berita *hoax* yang perlu diketahui sebagai antisipasi agar tidak termakan *hoax* yang sedang beredar, antara lain (Nasution, 2017): *Pertama,* terdapat anjuran untuk menyebarkan pesan tersebut ke orang lain. *Kedua*, penggunaan tata bahasa yang kurang sempurna. *Ketiga,* tidak adanya berita lain yang mendukung pesan tersebut. *Keempat,* pesan yang tersebar tidak logis, tidak konsisten dan bertentangan dengan akal sehat. *Kelima,* tidak menyebutkan kenyataan yang dapat dibuktikan dengan fakta-fakta. *Keenam,* pesan berantai atau pesan yang telah di*foward* berkali-kali. *Ketujuh,* pembuat *hoax* akan menghubungkan pesan tersebut dengan sumber resmi yang sebenarnya palsu seperti *hoax* tentang pemenang undian berhadiah dengan menyertakan alamat website yang terlihat seperti asli namun sebenarnya website palsu yang telah dibuat sendiri sebelumnya.

* 1. **Media Sosial**

Menurut *McGraw Hill Dictionary,* media sosial adalah segala bentuk media komunikasi interaktif yang memungkinkan terjadinya interaksi dua arah dan umpan balik. Media sosial dapat didefinisikan sebagai sarana yang digunakan oleh orang-orang untuk berinteraksi satu sama lain dengan cara menciptakan, berbagi serta bertukar informasi dan gagasan dalam sebuah jaringan dan komunikasi virtual. Dengan kata lain, media yang digunakan untuk menjadi sosial (Safko, 2012). Media sosial dapat digunakan untuk berbagi teks, gambar, suara dan video (Kotler & Keller, 2012). Berdasarkan berbagai penjelasan tersebut, dapat disimpulkan bahwa media sosial merupakan media komunikasi yang digunakan untuk melakukan interaksi satu sama lain dengan tujuan untuk berbagi dan bertukar informasi baik berupa teks, gambar, suara maupun video untuk menjadi makhluk sosial.

Media sosial dapat dikenali melalui beberapa karakteristik atau ciri-ciri yang dimiliknya, seperti adanya partisipasi pengguna untuk memberikan umpan balik terhadap konten di media sosial, adanya keterbukaan bagi pengguna untuk memberikan komentar dan sebagainya, adanya interaksi antar pengguna dan semua itu saling terhubung satu sama lain.

Pada saat ini, media sosial memiliki peran besar dalam membangun pola pikir dan perilaku dalam berbagai bidang kehidupan masyarakat. Hal tersebut tidak lepas dari berbagai fungsi yang ditawarkan dalam penggunaan media sosial. Beberapa fungsi dari media sosial, yaitu (Wijayanto, 2012):

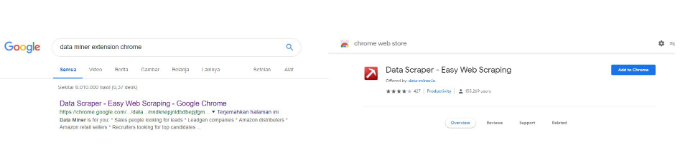
* 1. ***Web Scraping***

*Web scraping* adalah salah satu metode untuk melakukan pengambilan data. Cara kerja dari metode ini dengan mengekstraksi data yang berada di suatu website dan menganalisis data tersebut untuk digunakan bagi kepentingan lain (Turland, 2010). *Web scraping* berfokus pada pengambilan data dan ekstraksinya. *Web scraping* dapat diartikan sebagai bentuk penyalinan, dimana data spesifik dikumpulkan dan disalin dari suatu web dan disimpan di database lokal atau spreadsheet pusat yang dapat digunakan lagi di proses selanjutnya (Pereira & Vanitha, 2015). Pendapat lain menyatakan bahwa *web* scraping adalah sebuah proses yang memanfaatkan dokumen berbentuk website dibangun dengan bahasa *markup* seperti HTML atau XHTML berbentuk *semi-structured* yang berasal dari internet dan kemudian dianalisis untuk mendapatkan suatu informasi yang berguna untuk konteks lain (Nikhit, 2015).

Proses *web scraping* pertama kali dilakukan dengan cara manual yaitu dengan *copy paste* dari website ke tempat penyimpanan lokal. Namun, apabila data yang ingin diambil dalam jumlah banyak, maka cara ini kurang efektif karena memerlukan tenaga untuk menyalin dan juga memakan banyak waktu. Selain dengan cara manual, *web scraping* dapat dilakukan dengan otomatis. Beberapa cara untuk melakukan *scraping* secara otomatis dengan menggunakan *coding,* aplikasi ataupun *extension browser*. Sedangkan pada penelitian ini, metode yang digunakan untuk melakukan *scraping* yaitu dengan *extension browser* dari *google* yang bernama *Data Miner*.

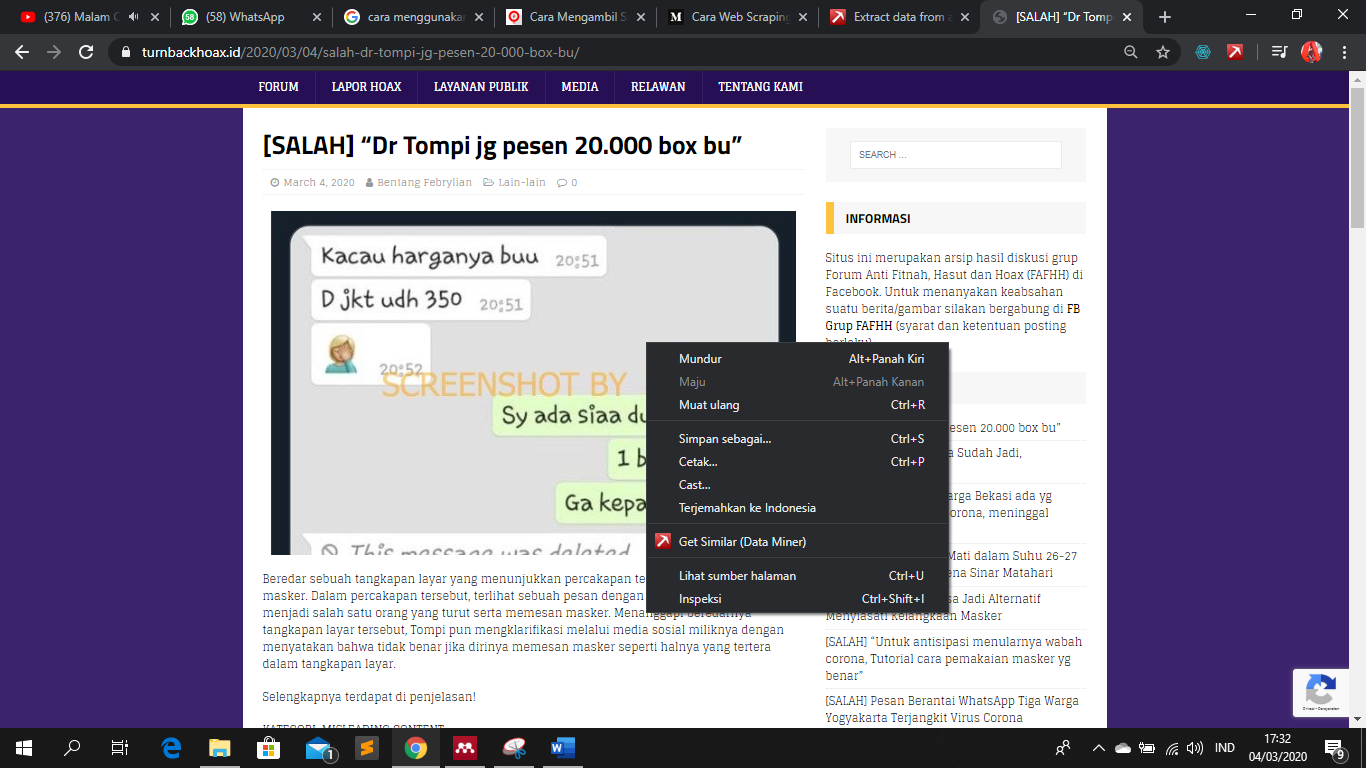
*Data miner* adalah salah satu *extension chrome* yang berfungsi untuk melakukan proses *scraping* data dari suatu website. *Data miner* akan melakukan *scraping* dengan mengekstrak data dari suatu website dan hasilnya disimpan pada *spreadsheet* atau excel serta dapat di*eksport* ke dalam bentuk file XLS, CSV, XLSX atau TSV (.xls, .csv, .xlsx, .tsv). Kelebihan dari *data miner* yaitu bisa digunakan dan dipahami oleh semua orang tanpa harus menguasai struktur HTML dari website, adanya fitur *auto pagination* untuk melanjutkan *scraping* ke halaman berikutnya dari website tersebut dan prosesnya *scraping*nya dilakukan secara otomatis. Apabila ingin menggunakan *data miner* untuk melakukan *scraping,* dapat dilakukan dengan cara sebagai berikut:

1. Tambahkan *extension Data Miner* terlebih dahulu di browser chrome. Cara menambahkan *data miner* dapat dilihat pada Gambar 2.1



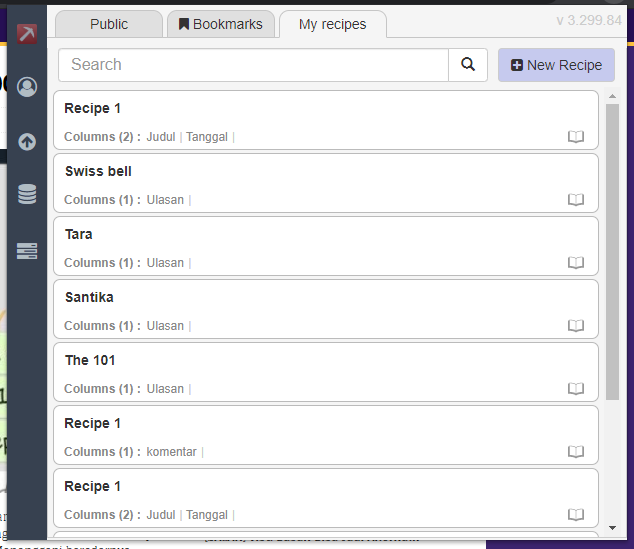
**Gambar 2.1** Tambah *Extension Data Miner*

1. Buka website yang datanya ingin di-*scraping*. Misalnya website turnbackhoax.id dan membuka salah satu beritanya. Kemudian klik kanan dan pilih *Get Similar (Data Miner)* seperti pada Gambar 2.2.



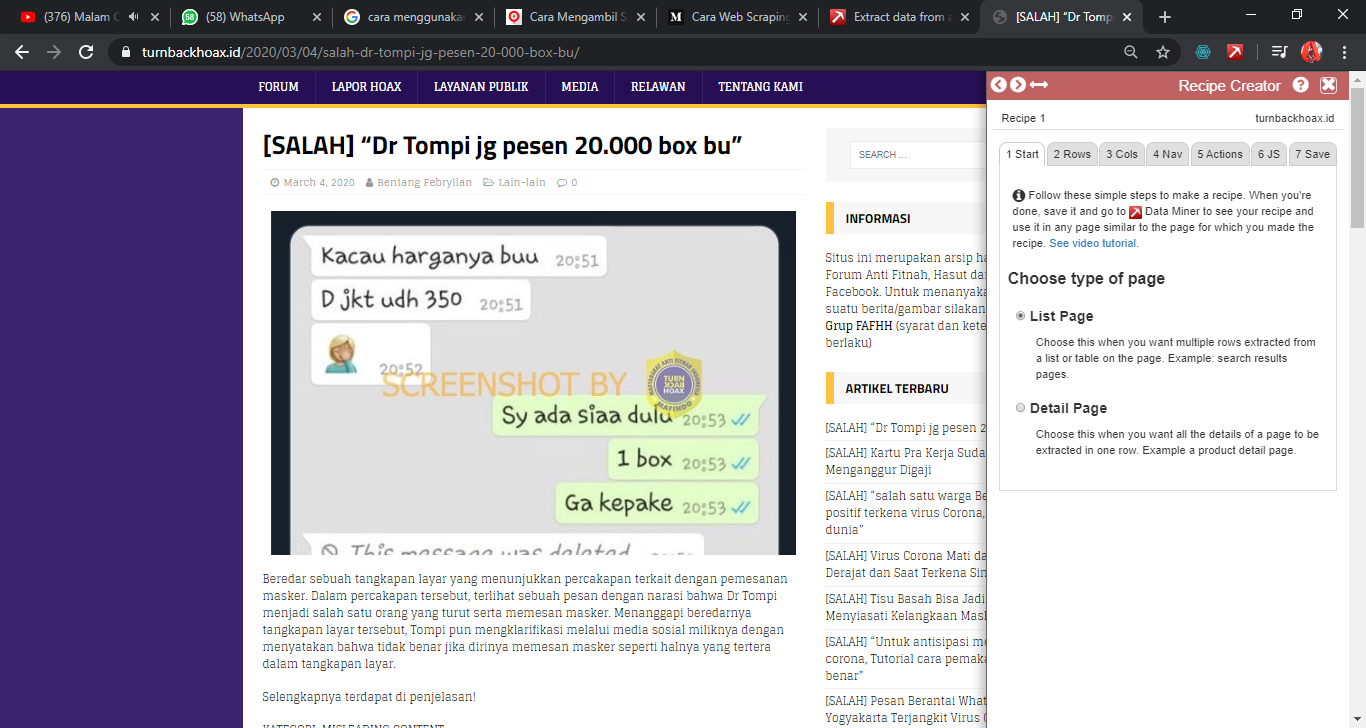
**Gambar 2.2** Proses Membuka *Data Miner*

1. Selanjutnya lakukan proses *sign in* dan klik *New Recipe* seperti pada Gambar 2.3.



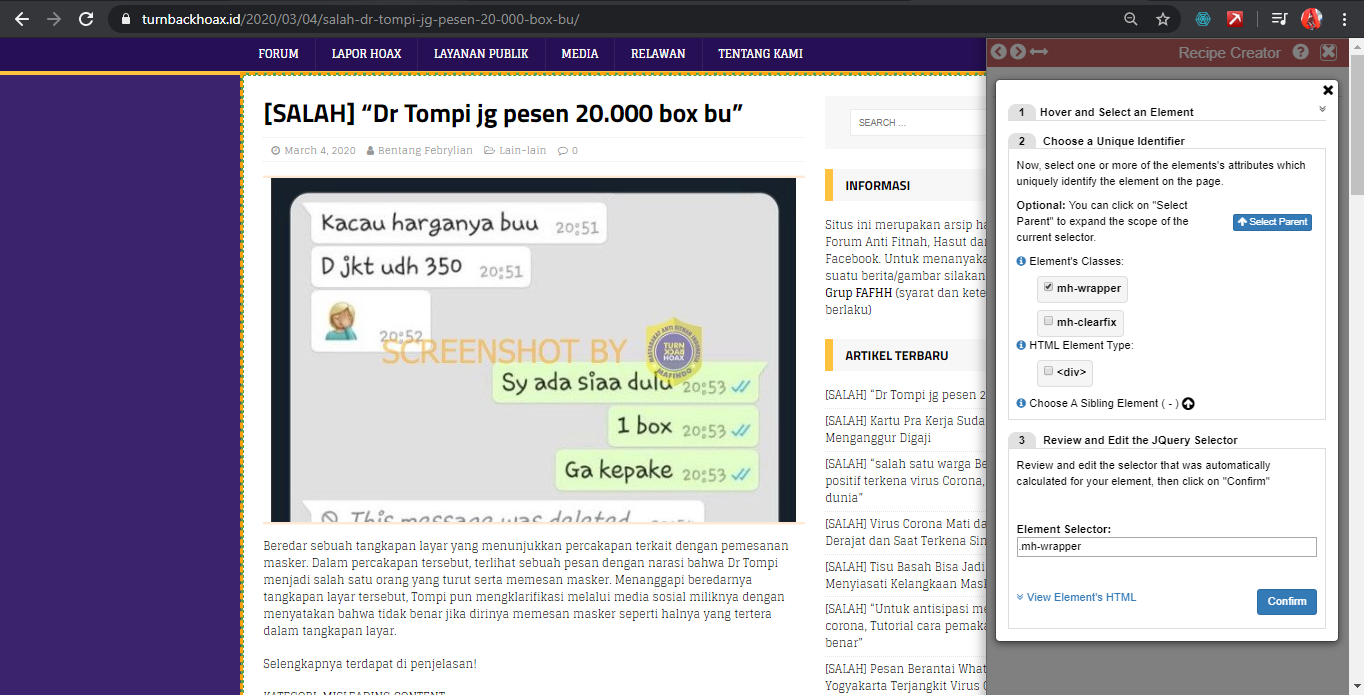
**Gambar 2.3** Proses Membuat *Recipe* Baru

1. Kemudian akan muncul tampilan seperti pada Gambar 2.4. Pada menu *start* pilih *list page* karena website yang akan di*scraping* memiliki data yang berbentuk *list* dan memiliki lebih dari satu halaman.



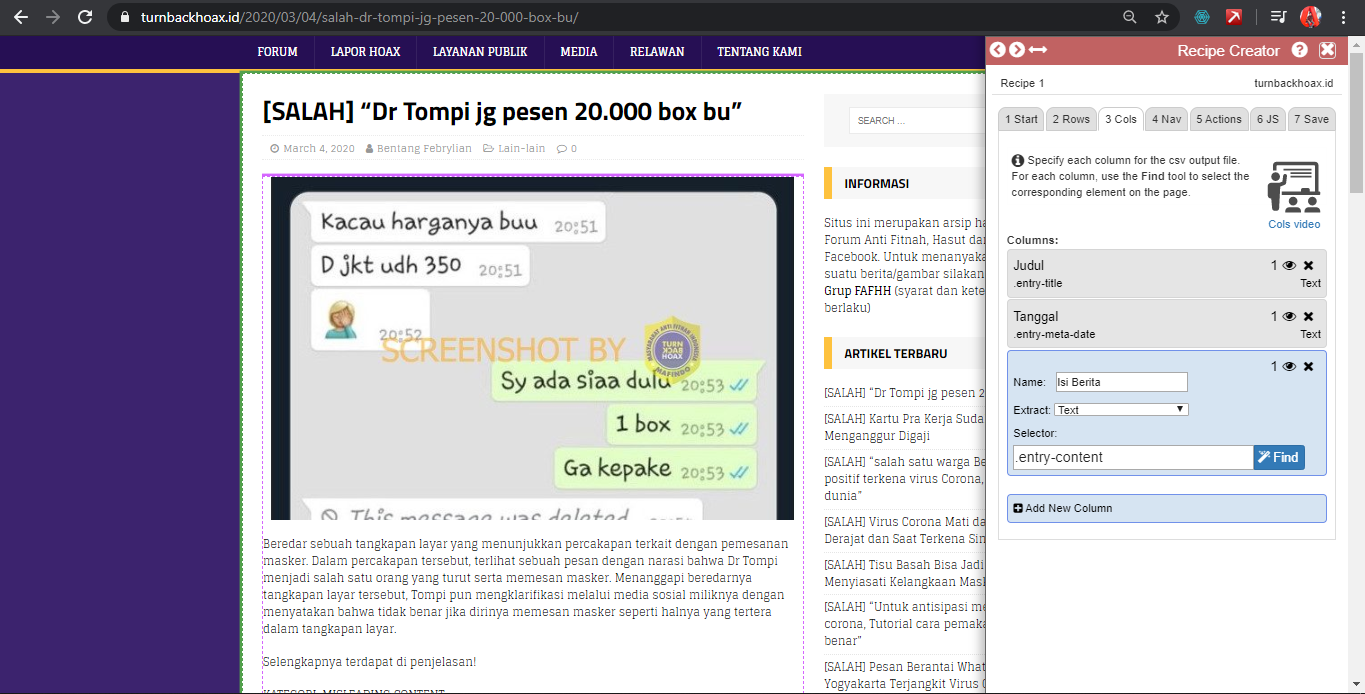
**Gambar 2.4** Pengaturan pada Menu *Start*

1. Setelah itu ke menu *Rows*. Menu *rows* berfungsi untuk memilih bagian halaman yang memuat semua informasi atau variabel data yang ingin dibutuhkan. Cara memilihnya yaitu dengan menekan tombol *find* lalu pilih jika blok berwarna kuning memuat semua data yang dibutuhkan. Untuk mengunci blok yang dipilih dengan menekan tombol *shift* pada *keyboard* kemudian centang pada elemen *class*. Maka akan tampil kotak hijau lalu klik ok. Proses ini dapat dilihat pada Gambar 2.5.



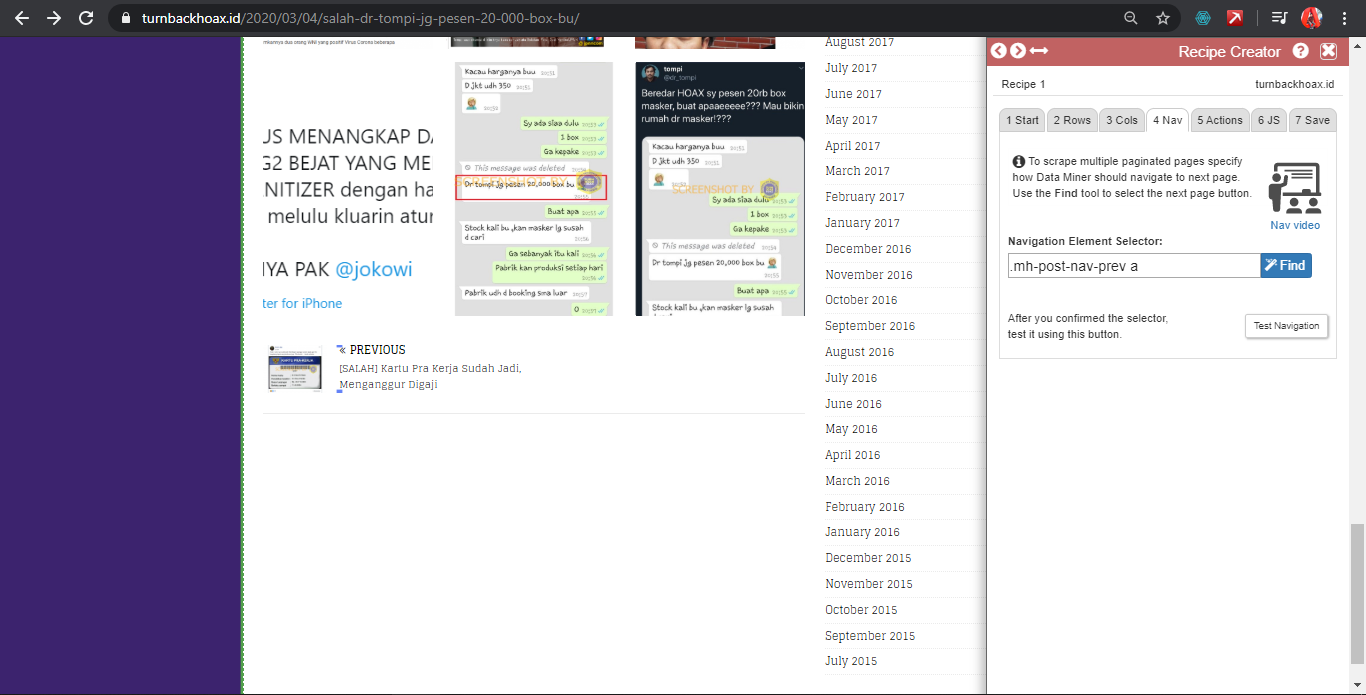
**Gambar 2.5** Proses Pemilihan Bagian yang akan di*scraping*

1. Menentukan setiap kolom atau variabel data yang ingin di*scraping.* Misalnya akan mengambil data judul berita, tanggal dan isi berita. Isiikan nama kolom sesuai dengan datanya dan *extract*nya bertipe teks. Lalu tambahkan sesuai dengan jumlah kolom yang diinginkan. Selanjutnya tekan tombol *find* seperti cara pada *rows*, bedanya pada bagian ini mencari letak dari masing-masing variabel. Apabila berhasil akan muncul jumlah data yang di*scrap* dari masing-masing variabelnya pada logo mata. Proses ini dapat dilihat pada Gambar 2.6.



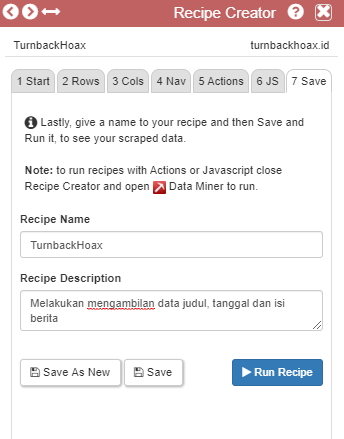
**Gambar 2.6** Proses Pemilihan Variabel Data

1. Berikutnya pengaturan di menu *Nav*. Menu ini bersifat opsional dan digunakan jika ingin mengambil data dari beberapa halaman baik halaman berikutnya atau halaman sebelumnya secara berulang terus-menerus. Cara penggunaannya sama seperti sebelumnya dengan nilai *find* mengarah pada link yang berfungsi untuk ke halaman sebelumnya atau berikutnya. Untuk lebih jelasnya dapat dilihat pada Gambar 2.7.



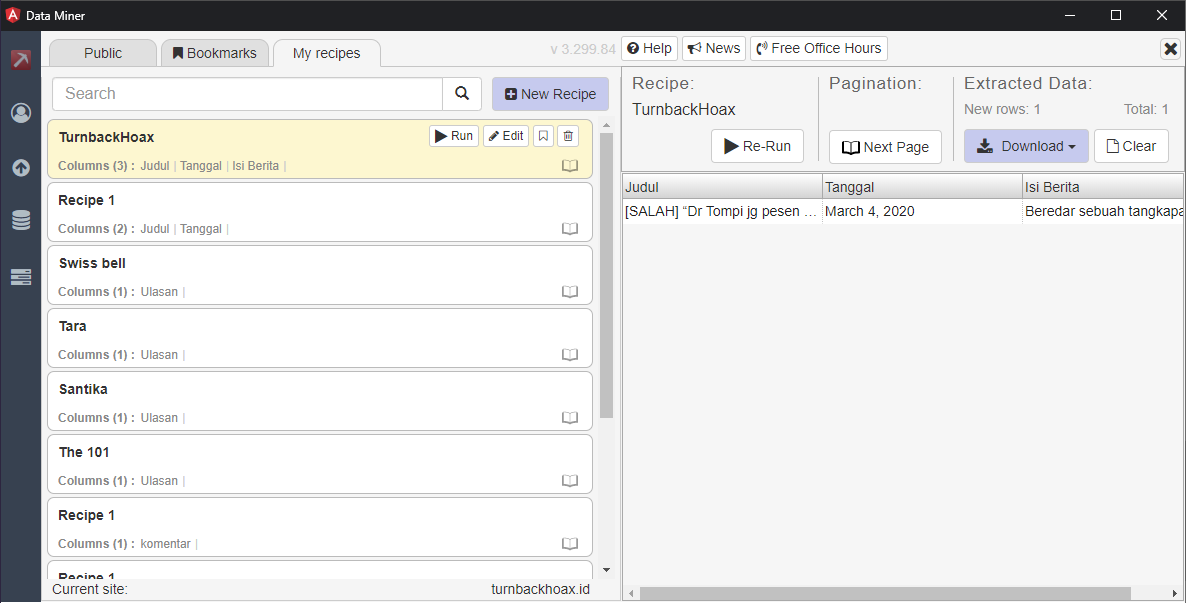
**Gambar 2.7** Penentuan *Navigator*

1. Tahap selanjutnya bisa langsung ke menu *Save*. Proses ini untuk menyimpan *recipe* sehingga dapat dibuka lagi di lain waktu. Berikan nama *recipe* dan deskripsinya untuk memperjelas keterangan dari setiap *recipe* seperti pada Gambar 2.8*.* Apabila sudah disimpan, klik tombol *Run Recipe* untuk melakukan proses *scraping*.



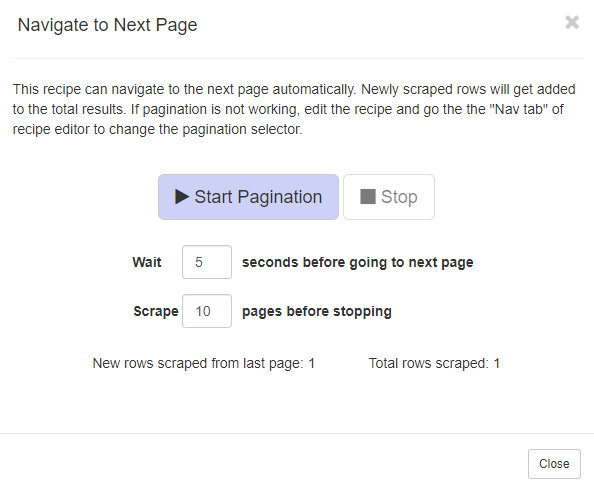
**Gambar 2.8** Proses Penyimpanan *Recipe*

1. Setelah *run recipe,* akan muncul hasil seperti pada Gambar 2.9 dengan data yang telah berhasil di *scraping*.



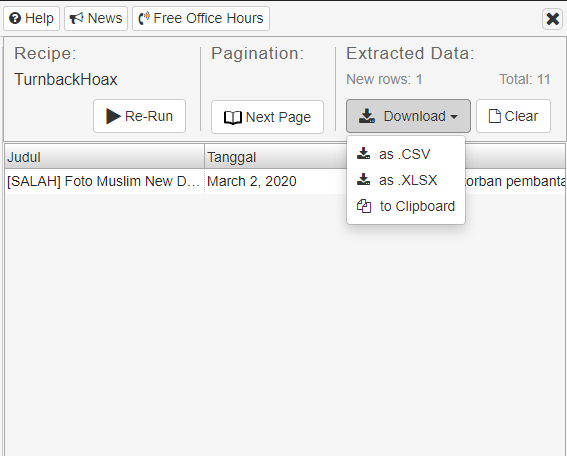
**Gambar 2.9** Hasil *Scraping*

10. Namun, hasil dari proses sebelumnya hanya melakukan *scraping* data yang ada di satu halaman. Kemudian jika ingi mengambil data di halaman berikutnya seperti pada menu *nav*, maka dapat mengklik *next page* pada menu *pagination*. Lalu akan muncul tampilan seperti pada Gambar 2.10 dan maksud dari nilai *wait* berarti jeda waktu untuk mengambil dari halaman satu ke halaman berikutnya dengan satuan detik sedangkan nilai *scrape* berarti jumlah halaman yang ingin di*scraping*.



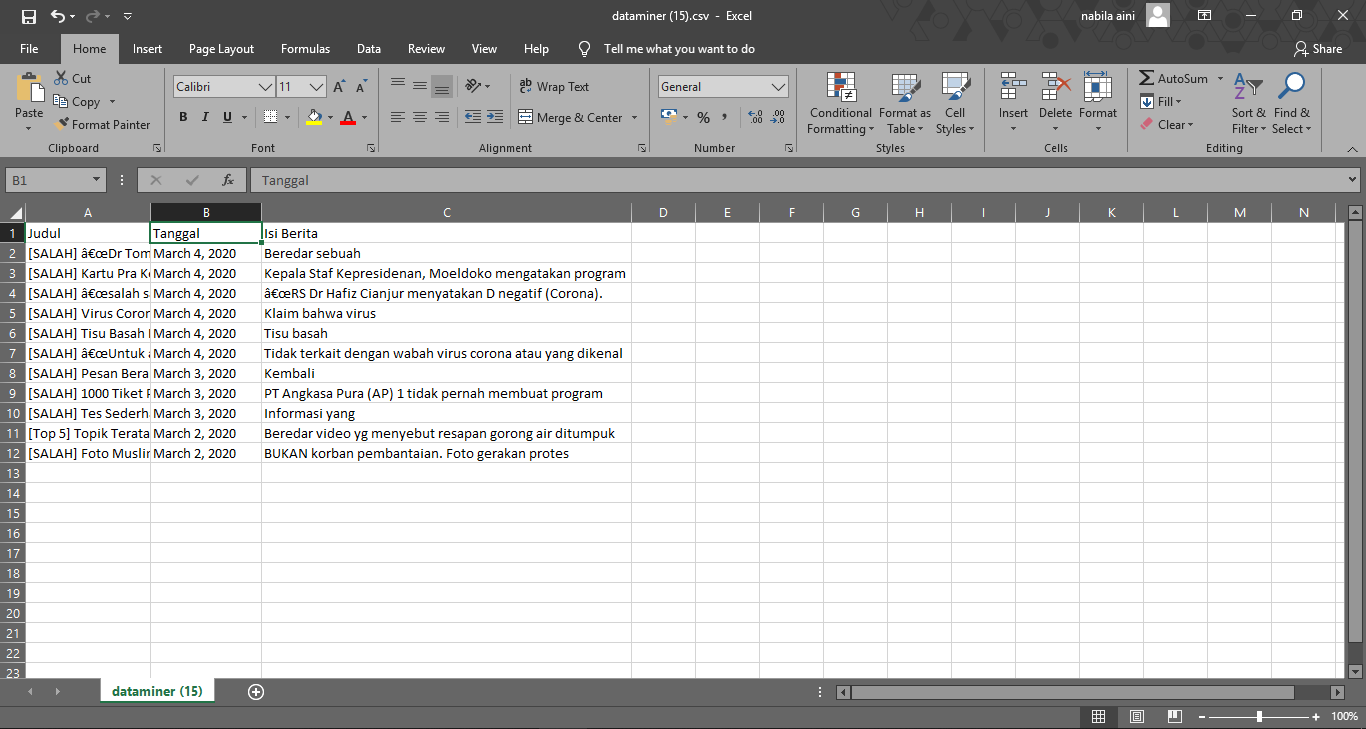
**Gambar 2.10** *Navigator* untuk Melanjutkan Proses *Scraping*

11. Tahap terakhir yaitu menyimpan data hasil dari *scraping* tadi. Cara untuk menyimpannya dengan mengklik tombol *download* pada menu *extracted data*. Kemudian pilih format file yang diinginkan. Caranya dapat dilihat pada Gambar 2.11.



**Gambar 2.11** Proses *Export* Data

12. Berikut hasil data dari proses *scraping* dengan *data miner* yang telah dieksport ke format file CSV (.csv) dapat dilihat pada Gambar 2.12.

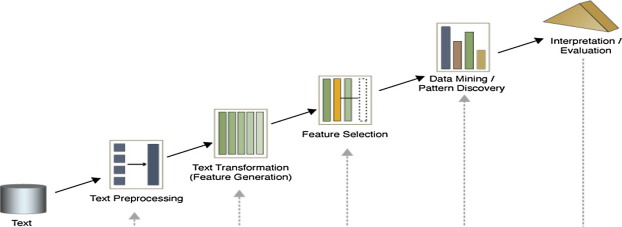


**Gambar 2.12** Hasil Data *Scraping*

* 1. ***Text Mining***

*Text mining* merupakan proses mencari untuk mencari pola pada kumpulan data-data. Proses pencarian pola pada *text mining* dilakukan oleh komputer untuk mendapatkan sesuatu hal yang baru dalam bentuk informasi atau menemukan kembali informasi yang tersirat secara implisit dengan cara melakukan ekstraksi informasi dari sumber data teks yang berbeda-beda (Feldman & Sanger, 2007). *Text mining* dapat menganalisis dokumen, mengelompokkan dokumen berdasarkan kata yang terkandung didalamnya serta dapat menemukan kesamaan antar dokumen sehingga dapat mengetahui hubungan antar variabelnya.

Penggunaan *text mining* biasanya untuk menganalisis data teks yang semi-terstruktur ataupun data yang tidak terstruktur. Maka dari itu, *text mining* bertanggung jawab untuk merubah data yang awalnya semi-terstruktur atau tidak terstruktur menjadi data yang terstruktur. Berbagai jenis permasalahan yang dapat menggunakan *text mining* seperti klasifikasi, *clustering, information extraction* dan *information retreival* (Berry & Kogan, 2010)*.* Dari berbagai permasalahan tersebut dapat diaplikasikan pada suatu kasus, seperti klasifikasi artikel berita, analisis sentimen, penyaringan spam email dan lain sebagainya. Beberapa tahapan yang biasanya dilakukan untuk melakukan *text mining,* yaitu *text preprocessing, text transformation, feature selection, pattern discovery* dan *interpretation*. Tahapan *text mining* lebih jelasnya dapat dilihat pada Gambar 2.13.



**Gambar 2.13** Tahapan *Text Mining* (Hashimi, Hafez, & Mathkour, 2015)

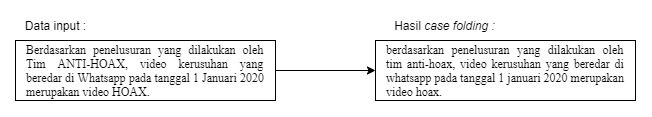
* 1. ***Text Preprocessing***

*Text preprocessing* merupakan salah satu tahapan yang penting pada proses *text mining*. *Text preprocessing* adalah proses untuk membersihkan dan mempersiapkan data sebelum dilakukannya proses klasifikasi (Haddi, Liu, & Shi, 2013). Penggunaan *preprocessing* dikarenakan data yang digunakan pada proses *mining* tidak selamanya dalam kondisi ideal untuk diproses. Tidak jarang data yang akan digunakan memiliki berbagai permasalahan yang dapat mempengaruhi hasil dari proses *mining* itu sendiri. Permasalahan-permasalahan yang sering muncul pada data yang akan diproses seperti *missing value, data redundant, outliers* ataupun format data yang tidak sesuai dengan sistem. Maka dari itu, proses ini akan mengubah bentuk data yang awalnya semi-terstruktur ataupun tidak terstruktur menjadi data yang terstruktur.

Berdasarkan penjelasan tersebut, dapat disimpulkan bahwa *text preprocessing* bertujuan untuk mendapatkan bentuk data yang siap diolah atau diproses. Tahapan *text preprocessing* yang akan digunakanmeliputi *case folding, remove punctuation, remove number, tokenizing, stopword removal* dan *stemming*.

* + 1. ***Case Folding***

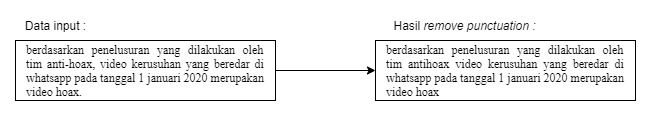
Tahap *case folding* adalah proses merubah semua huruf pada dokumen menjadi huruf kecil (*lowercase*). Huruf yang diterima hanya ‘a’ sampai dengan ‘z’. Perubahan huruf ini bertujuan untuk membentuk suatu standar pada data. Misalnya terdapat data yang berisi “Berita HOAX” akan menjadi “berita hoax”. Contoh lain dari proses *case folding* dapat dilihat pada Gambar 2.14.



**Gambar 2.14** Proses *Case Folding*

* + 1. ***Remove Punctuation***

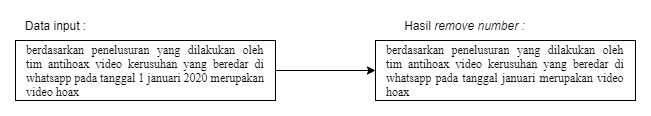
Tahap *remove punctuation* merupakan proses untuk menghapus tanda baca yang terdapat pada suatu data. Pada tahap ini, tanda baca di anggap sebagai *delimiter.* Penghapusan tanda baca pada suatu data akan mengurangi beban pemrosesan. Beberapa contoh tanda baca seperti titik (.), koma (,), tanda tanya (?), tanda seru (!) dan lainnya. Misalnya terdapat data yang berisi “Cek Fakta: Virus Corona tidak bisa masuk Indonesia?” akan berubah menjadi “Cek Fakta Virus Corona tidak bisa masuk Indonesia”. Contoh lain dari proses *remove punctuation* dapat dilihat pada Gambar 2.15.



**Gambar 2.15** Proses *Remove Punctuation*

* + 1. ***Remove Number***

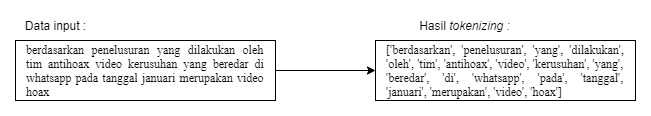
*Remove number* merupakan tahapan yang hampir sama dengan *remove punctuation*. Bedanya pada bagian objek yang akan dihapus. Pada *remove punctuation* menghapus tanda baca, sedangkan *remove number* akan menghapus semua angka pada suatu data. Angka akan dihapus karena dianggap tidak memiliki arti dan termasuk *delimiter*. Misalnya ada data yang berisi “terdapat 2 akun media sosial yang menyebar berita hoax” akan diubah menjadi “terdapat akun media sosial yang menyebar berita hoax”. Contoh lain penggunaan dari *remove number* dapat dilihat pada Gambar 2.16.



**Gambar 2.16** Proses *Remove Number*

* + 1. ***Tokenizing***

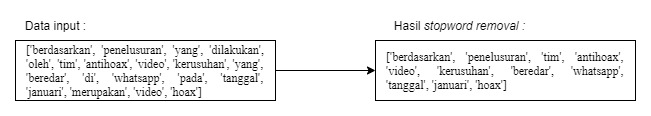
Tahap *tokenizing* atau tokenisasi adalah proses pemisahan teks dari paragraph atau kalimat berdasarkan per kata. Proses *tokenizing* biasanya berpatokan pada karakter *whitespace* seperti spasi, enter ataupun tab sebagai pemisah atau pemotong antar kata. Tiap kata yang akan dihasilkan pada proses ini akan disebut sebagai token*.* Contoh tokenisasi dari kalimat ”komparasi metode deep learninguntuk klasifikasi berita hoax” akan menghasilkan 8 token, yaitu: “komparasi”, “metode”, “deep”, “learning”, “untuk”, “klasifikasi”, “berita”, dan “hoax”. Contoh lain yang dapat memperjelas proses *tokenizing* dapat dilihat pada Gambar 2.17.



**Gambar 2.17** Proses *Tokenizing*

* + 1. ***Stopword Removal***

*Stopword removal* adalah proses penghapusan kata *stopword*. *Stopword* adalah kata-kata umum yang sering muncul pada suatu dokumen atau data namun tidak memiliki makna (tidak berkaitan dengan tema dokumen). *Stopword* biasanya berupa kata ganti orang atau kata penghubung. Beberapa contoh dari kata *stopword* seperti ‘aku’, ‘kamu’, ‘yang’, ‘di’, ‘dan’, ‘atau’, ‘tetapi’ dan lain sebagainya. Jadi, *stopword removal* merupakan proses pembuangan kata-kata kurang penting (*stopword*) dari token yang telah dihasilkan pada tahap sebelumnya. Pembuangan *stopword* tidak akan mengubah makna atau isi dari suatu data. Bahkan dengan menghapus *stopword* dapat mengurangi ukuran index, waktu pemprosesan dan *noise* dari suatu data. Contoh penerapan dari *stopword removal* dengan inputan hasil dari proses *tokenizing* dapat dilihat pada Gambar 2.18.



**Gambar 2.18** Proses *Stopword Removal*

* + 1. ***Stemming***

*Stemming* adalah suatu proses untuk mentransformasi kata-kata yang terdapat pada suatu dokumen menjadi kata dasar (*root* word). Biasanya proses yang dilakukan dengan menghapus awalan, akhiran maupun sisipan. Hal ini bertujuan untuk mengurangi varian kata dengan makna yang hampir sama pada suatu dokumen. Selain itu, proses *stemming* dapat meningkatkan kinerja pengambilan data karena jumlah indeks dengan istilah yang berbeda menjadi berkurang. Contoh proses transformasi kata pada *stemming* pada kata “menghasilkan” menjadi “hasil”. Terdapat beberapa algoritma yang telah dikembangkan untuk melakukan proses *stemming*. Penggunaan algoritma *stemming* harus disesuaikan dengan bahasa yang digunakan. Algoritma yang digunakan untuk teks dokumen yang menggunakan bahasa Indonesia akan berbeda dengan teks yang berbahasa Inggris. Pada teks berbahasa Indonesia dapat menggunakan beberapa algoritma, salah satunya Algoritma Nazief dan Adriani.

Algoritma Nazief dan Adriani adalah algoritma *stemming* untuk bahasa Indonesia. Algoritma ini memiliki beberapa aturan morfologi untuk merubah suatu kata menjadi kata dasar yaitu dengan menghilangkan *affiks* (awalan, imbuhan dan lain-lain) dan kemudian dicocokkan dengan kamus kata dasar. Kamus kata dasar merupakan hal yang paling utama pada algoritma ini karena semakin banyak kamus kata dasarnya maka semakin tinggi pula akurasinya. Algoritma ini memiliki tahap-tahap sebagai berikut (Nazief & Adriani, 1996):

1. Cari kata yang akan di-*stem* di dalam kamus, jika data tersebut ditemukan maka kata tersebut adalah kata dasar dan algoritma berhenti. Jika tidak ada maka lanjutkan ke langkah-2.

2. Hilangkan *inflectional suffix* (imbuhan infleksional) yaitu (“-lah”, ”-kah”, “-tah”, “-ku”, “-mu”, “-nya”).

3. Hapus *derivation suffix* (imbuhan turunan) yaitu (“-i”, “-an”, atau “-kan”). Jika kata ditemukan di kamus, maka algoritma berhenti. Jika tidak maka ke langkah-3a.

a. Jika “-an” telah dihapus dan huruf terakhir dari kata tersebut adalah “-k” maka “-k” juga ikut dihapus. Jika kata tersebut ditemukan dalam kamus maka algoritma berhenti. Jika tidak ditemukan maka lakukan langkah-3b.

b. Akhiran yang dihapus (“-i”, “-an” atau “-kan”) dikembalikan dan lanjut ke langkah-4.

4. Hapus *derivation prefix* (awalan turunan) yaitu (“be-“, “di-“, “ke-“, “me-“, “pe-“, “se-“, “te-“). Jika pada langkah 3 ada *suffix* yang dihapus maka pergi ke langkah-4a, jika tidak maka pergi ke langkah-4b.

a. Periksa tabel kombinasi awalan-akhiran yang tidak diizinkan. Jika ditemukan maka algoritma berhenti, jika tidak pergi ke langkah-4b. Kombinasi awalan dan akhiran yang tidak diijinkan dapat dilihat pada Tabel 2.1.

**Tabel 2.1** Kombinasi Awalan dan Akhiran yang Tidak Diijinkan (Nazief & Adriani, 1996)

|  |  |
| --- | --- |
| **Awalan** | **Akhiran yang tidak diijinkan** |
| be- | -i |
| di- | -an |
| ke- | -i, -kan |
| me- | -an |
| se- | -i, -kan |

b. Untuk i=1 sampai 3, tentukan tipe awalan kemudian hapus awalan. Jika kata dasar belum ditemukan juga lakukan langkah-5, jika sudah maka algoritma berhenti. Proses ini harus dilihat terlebih dahulu aturan peluruhan kata pada Tabel 2.2.

**Tabel 2.2** Aturan Peluruhan Kata Dasar (Nazief & Adriani, 1996)

|  |  |  |
| --- | --- | --- |
| **Aturan** | **Awalan** | **Peluruhan** |
| 1 | berV. . . | ber-V.. | be-rV.. |
| 2 | Belajar | bel-ajar |
| 3 | berClerC2 | be-ClerC2.. dimana C1!= {‘r’|’l’} |
| 4 | terV. . . | ter-V.. | te-rV.. |
| 5 | terCer. . . | terCer. . . dimana C!=’r’ |
| 6 | teClerC2 | te-CleC2. . . dimana C1!=’r’ |
| 7 | me{I|r|w|y}V. . . | me-{I|r|w|y}V. . . |
| 8 | mem{b|fv}. . . | mem-{b|f|v}. . . |
| 9 | mempe. . . | m-pe. . |
| 10 | mem{r|V|V}. . . | me-m{r|V|V}. . . | me-p{r|V|V}. . . |
| 11 | men{c|d|j|z}. . . | Men-{c|d|j|z}. . . |
| 12 | menV. . . | Me-nV. . . | me-tV. . . |
| 13 | meng{g|h|q|k}. . . | Meng-{g|h|q|k}. . . |
| 14 | mengV. . . | Meng-V. . . | meng-kV. . . |
| 15 | mengeC. . . | Meng-C. . . |
| 16 | menyV. . . | Me-ny. . . | men-sV. . . |
| 17 | memV. . . | Mem-pV. . . |
| 18 | pe{w|y}V. . . | Pe-{w|y}V. . . |
| 19 | perV. . . | Per-V. . . | pe-rV. . . |
| 20 | pem{b|f|v}. . . | Pem-{b|f|v}. . . |
| 21 | pem{rV|V}. . . | Pe-m{rV|V}. . . | pe-p{rV|V}. . . |
| 22 | pen{c|d|j|z}. . . | Pen-{c|d|j|z}. . . |
| 23 | penV. . . | Pe-nV. . . | pe-tV. . . |
| 24 | peng{g|h|q}. . . | Peng-{g|h|q} |
| 25 | pengV. . . | Peng-V | peng-kV |
| 26 | penyV. . . | Pe-nya | peny-sV |
| 27 | pelV. . . | Pe-IV. . .; kecuali untuk kata “pelajar” |
| 28 | peCP. . . | Pe-CP. . . dimana C!={r|w|y|I|m|n} dan P!=’er’ |
| 29 | perCerV. . . | Per-CerV. . . dimana C!={r|w|y|I|m|n} |

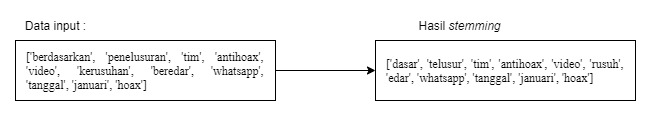
Tipe awalan ditentukan melalui langkah-langkah sebagai berikut:

1. Jika awalannya adalah: “di-”, “ke-“ atau “se-“ maka tipe awalannya secara berturut-turut adalah “di-“, “ke-“ atau “se-”.
2. Jika awalannya adalah: “te-“, “me-“, “be-“ atau “pe-“ maka dibutuhkan sebuah proses tambahan untuk menentukan tipe awalannya.
3. Jika dua karakter pertama bukan “di-“, “ke-“, “se-“, “te-“, “be-“, “me-“ atau “pe-“ maka berhenti.
4. Jika tipe awalan adalah “none” maka berhenti. Hapus awalan jika ditemukan.

5. Lakukan *recording*. Proses *recording* dilakukan dengan menambah karakter *recording* di awal kata yang dipenggal dengan mengacu pada Tabel 2.5.

6. Jika semua langkah telah selesai tetapi tidak juga berhasil maka kata awal diasumsikan sebagai kata dasar. Proses selesai.

Sebagai contoh dari proses *stemming,* misalnya terdapat kata “menghasilkan” menjadi “hasil” atau kata “berasal” akan menjadi “asal”. Contoh lain untuk memperjelas proses *stemming* dapat dilihat pada Gambar 2.19.



**Gambar 2.19** Proses *Stemming*

* 1. **Klasifikasi**

Klasifikasi adalah proses untuk menemukan sebuah model atau fungsi yang menggambarkan dan membedakan antar kelas data. Klasifikasi didefinisikan sebagai pengkategorian dokumen secara otomatis ke dalam satu atau lebih kelas yang telah ditentukan berdasarkan isinya (Sebastiani, 2002). Dokumen-dokumen yang memiliki isi yang sama atau relevan akan dikelompokkan pada ketegori yang sama. Dengan kata lain bahwa klasifikasi merupakan pemberian ketegori yang telah didefinisikan kepada dokumen yang belum memiliki kategori (Goller, 2000). Tujuan dari klasifikasi yaitu untuk memperkirakan kelas dari suatu objek yang belum diketahui kelasnya (Gaikwad, Chaugule, & Patil, 2014). Selain itu, akan memudahkan mencari suatu informasi dengan isi yang relevan.

Klasifikasi memiliki dua proses yaitu proses *training* dan *testing*. Proses *training* adalah proses untuk melatih algoritma yang digunakan dengan dokumen yang telah memiliki kategori. Proses *training* akan menghasilkan suatu model yang nantinya dapat digunakan untuk melakukan prediksi kategori dari suatu data. Sedangkan, *proses testing* yaitu proses untuk menguji algoritma dengan data yang belum memiliki kategori untuk memprediksi data tersebut termasuk kategori yang mana. Pada proses *testing* akan diketahui tingkat akurasi dari model yang telah dibangun dari proses *training*.

* 1. ***Word Embedding***

*Word embedding* adalah proses untuk mengubah suatu kata menjadi sebuah vektor yang terdiri dari kumpulan angka. Jumlah angka yang dihasilkan dari proses *word embedding* disebut dimensi. Penggunaan *word embedding* biasanya untuk algoritma *deep learning,* karenakan algoritma *deep learning* tidak dapat memproses data dalam bentuk *string,* maka dari itu harus dikonversi ke dalam bentuk angka terlebih dahulu. Contoh sederhana untuk merubah teks menjadi vektor angka dengan menggunakan *one-hot encoding*. Misalnya ada kalimat berisi “Indonesia adalah negara berkembang” maka akan dirubah menjadi vektor untuk masing masing kata dimana posisi kata direpresentasikan dengan nilai 1. Vektor yang merepresentasikan kata “Indonesia” adalah [1,0,0,0], kata “adalah” dengan vektor [0,1,0,0], “negara” direpresentasikan dengan [0,0,1,0] dan kata “berkembang” menjadi [0,0,0,1]. Selain menggunakan *one-hot encoding,* ada metode lain yang sering digunakan untuk melakukan *word embedding* yaitu *word2vec*.

*Word2vec* merupakan metode yang digunakan untuk melakukan transformasi dari teks menjadi suatu vektor angka. Proses *word embbeding* dengan metode *word2vec* menggunakan konsep *neural network* yang memetakan kata ke variabel target (Abdullah, 2018). Sama halnya dengan *neural network*, proses yang dilakukan dengan metode *word2vec* ini menggunaka *weight* sebagai representasi vektor kata. *Word2vec* akan mengetahui hubungan semantik/sintatikal antar kata diruang vektor. Pada implementasinya, *word2vec* memiliki dua teknik untuk melakukan proses *word embedding* yaitu *Continous Bag of Words* dan *Skip-Gram Model*. Misalnya ingin membuat model dari suatu data training yang terdiri nilai input berupa kata “king” dan “brave” maka hasilnya adalah “man”. Alur kerja yang dilakukan untuk membuat model tersebut apabila menggunakan *continous bag of words* adalah sebagai berikut:

1. Lakukan proses transformasi pada layer input dan layer output (target) menggunakan *one-hot encoding* sehingga vektor yang dihasilkan berukuran . Dimana nilai adalah jumlah kata dari hasil *tokenizing*. Misalnya pada contoh ini memiliki 6 kata maka matriks input berukuran dimana dan 2 adalah kata “king” dan “brave”. Bentuk matriks input yang dihasilkan dapat dilihat pada Formula 2.1.

(2.1)

Sedangkan matriks dari layer output berukuran seperti pada Formula 2.2.

(2.2)

2. Tahap selanjutnya yaitu melakukan pembobotan. Proses pembobotan dilakukan dua kali yaitu di antara *input layer* dengan *hidden layer* dan di antara *hidden layer* dengan *output layer*. Matriks *input hidden layer* berukuran dan matriks dari *output hidden layer* berukuran . Dimana nilai N merupakan jumlah *neuron*. Misalnya jumlah *neuron*nya adalah 4 maka matriks *input hidden layer* memiliki ukuran seperti pada Formula 2.3.

(2.3)

Sedangkan ukuran matriks dari *output hidden layer* adalah dapat dilihat pada Formula 2.4.

(2.4)

3. Kemudian *input layer* dikalikan dengan *weight* dari *input hidden layer*. Hasil dari proses ini disebut dengan *hidden activation*. Berikut persamaan untuk menghasilkan *hidden activation* pada Formula 2.5. Pada iterasi pertama, bobot akan diberikan secara random.

(2.5)

Setelah itu, cari nilai rata-rata dari vektor *hidden activation*. Hasil rata-rata *hidden activation* pada Formula 2.6.

(2.6)

4. Lakukan proses perkalian antara *hidden activation* dengan *hidden-output weight* untuk memperoleh *output*. Proses perkalian tersebut seperti pada Formula 2.7.

(2.7)

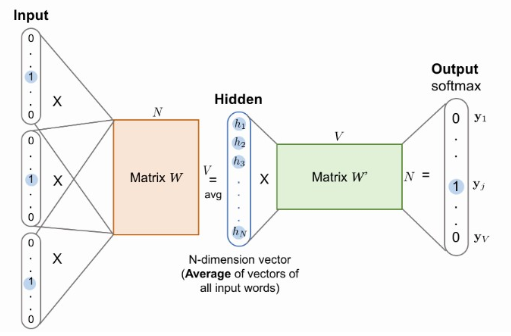
5. Setelah mendapat nilai *output,* maka *output* tersebut ditransformasikan menggunakan fungsi untuk mendapatkan nilai probabilitasnya. Berikut hasil probabilitas dari *output* pada Formula 2.8.

(2.8)

6. Hitung nilai *error* antara *output* dengan target kata. Setelah itu dilakukan *backpropagation* untuk *re-adjust weight*nya untuk memperbaiki nilainya. Cara menghitung yaitu matriks target dikurang matriks *output* seperti pada Formula 2.9.

(2.9)

Berdasarkan alur kerja tersebut, maka dapat diilustrasikan proses dari teknik *continous bag of words* seperti pada Gambar 2.20.



**Gambar 2.20** (Abdullah, 2018)

Sedangkan apabila menggunakan teknik *Skip-Gram Model,* alur kerja yang dilakukan sebagai berikut (Abdullah, 2018):

1. Lakukan transformasi dari nilai input menjadi berukuran , matriks *input hidden weight* berukuran dimana N merupakan jumlah *neuron* dan ukuran matriks dari *hidden output* adalah .

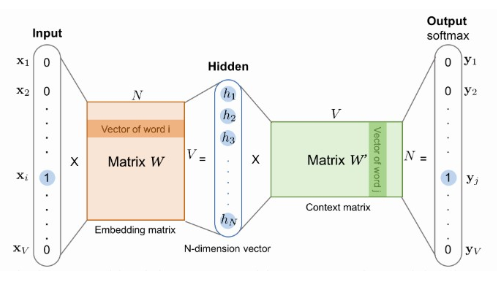
2. C adalah jumlah konteks katanya.

3. Kemudian nilai matriks dari *hidden activation*  dikalikan dengan *weight* yang dilakukan di antara *hidden layer* dan *output layer* untuk menghasilkan *output* dari prediksi.

4. Hasil dari proses sebelumnya berupa nilai *output* ditransformasikan menggunakan fungsi untuk mendapatkan nilai probabilitasnya.

5. Hitung *error* atau selisih antara nilai *output* dengan target. Kemudian dilakukan proses *backpropagation* untuk *re-adjust weight*nya.

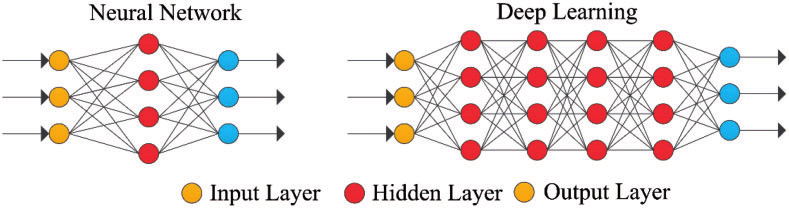
Alur kerja dari *skip-gram model* lebih jelasnya dapat dilihat ilustrasi pada Gambar 2.21



**Gambar 2.21** (Abdullah, 2018)

* 1. ***Deep Learning***

*Deep learning* adalah salah satu bidang dari pembelajaran mesin (*machine learning*) yang berbasis jaringan syaraf tiruan atau *neural network*. Sama halnya dengan *neural network, deep learning* juga akan mengajarkan kepada komputer untuk melakukan tindakan seperti manusia, dimana tindakan atau keputusan tersebut berdasarkan hasil belajar dari contoh yang diberikan. *Deep learning* merupakan sebuah pendekatan dalam penyelesaian masalah pada sistem pembelajaran komputer yang menggunakan konsep hirarki (Goodfellow, Bengio, & Courville, 2016). Hirarki pada *deep learning* terdiri dari banyaknya struktur dan jumlah jaringan syaraf pada algoritmanya hingga ratusan lapisan yang setiap lapisannya memiliki tanggung jawabnya masing-masing. Secara umum, lapisan dari *deep learning* terdiri dari tiga bagian yatu *input layer, hidden layer* dan *output layer*. Pada bagian *hidden layer* ini lah yang dapat dibuat berlapis-lapis untuk menemukan komposisi algoritma yang tepat agar meminimalisir *error* pada *output* (Primas, 2017). Arsitektur dari *deep learning* dapat diilustrasikan seperti pada Gambar 2.22.

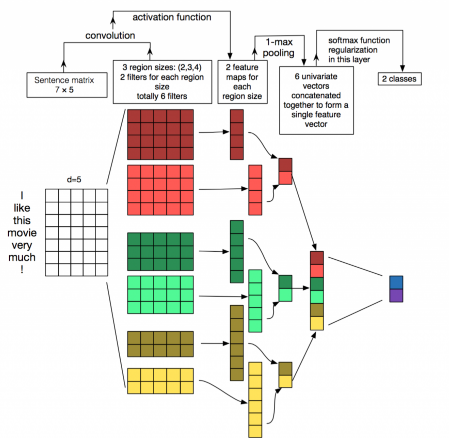


**Gambar 2.22** Diagram *Deep Learning* (Xing & Du, 2018)

*Deep learning* dapat digunakan untuk menangani berbagai permasalahan dengan data berskala besar seperti *computer vision, speech recognition* dan *natural language processing*. Kelebihan dari *deep learning* yaitu memiliki *feature engineering* yang berfungsi untuk mengekstrak pola yang penting dari suatu data sehingga dapat memudahkan model untuk membedakan suatu kelas. Model merupakan hasil dari proses pelatihan pada *deep learning* yang nantinya akan digunakan untuk melakukan prediksi. Algoritma yang digunakan pada *feature engineering* harus disesuaikan dengan kumpulan data dan jenis data yang digunakan. Pemilihan algoritma tersebut berdampak pada baik atau buruknya model untuk melakukan prediksi. Ada beberapa algoritma yang menerapkan konsep dari *deep learning* untuk melakukan *feature engineering* antara lain *Convolutional Neural Network* (CNN), *Recurrent Neural Network* (RNN) dan *Hierarchical Attention Network* (HAN).

* 1. ***Convolutional Neural Network* (CNN)**

*Convolutional neural network* (CNN) adalah salah satu algoritma dari *deep learning*. CNN merupakan jaringan syaraf yang dikhususkan untuk mengolah data yang memiliki grid. Sebenarnya, CNN merupakan pengembangan dari *Multilayer Perceptron* (MLP) yang di desain untuk mengolah data dua dimensi. Namun, ada beberapa penelitian yang menerapkan algoritma CNN pada *Natural Language Processing* (NLP) dengan hasil yang cukup memuaskan. Jadi, selain digunakan untuk mengolah data dua dimensi, CNN juga dapat digunakan untuk mengolah data satu dimensi termasuk melakukan klasifikasi teks (Razi, 2017) . Konsep dari CNN 1D tidak jauh berbeda dari *neural network* biasa dimana terdiri dari *neuron* yang memiliki *weight, bias* dan *activation function*. Secara garis besar, arsitektur dari CNN 1D dibagi menjadi beberapa tahapan yaitu *input layer, convolutional layer, activation layer, pooling layer* dan *fully connected layer.* Arsitektur dari CNN 1D dapat dilihat pada Gambar 2.23:



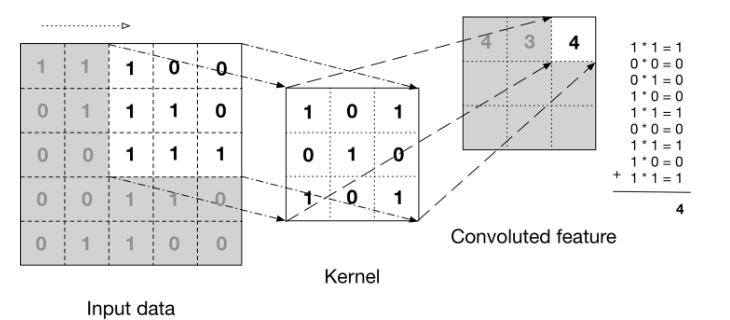
**Gambar 2.23** Arsitektur *Convolutional Neural Network* (Zhang & Wallace, 2015)

* + 1. ***Input Layer***

Layer input adalah layer pertama pada algoritma CNN. Layer ini berfungsi untuk menampung data yang menjadi inputan. Data yang ditampung berupa nilai vektor yang merepresentasikan dari masing-masing kata yang merupakan hasil dari proses *word embedding*. Ukuran layer ini sesuai dengan banyaknya kata dikali dengan panjang vektor atau dimensi vektor. Misalnya terdapat data yang terdiri dari 5 kata dan masing-masing kata direpresentasikan dengan 100 dimensi vektor, maka layer input berukuran .

* + 1. ***Convolution Layer***

*Convolution layer* adalah proses utama dari CNN. Pada tahap ini, *convolutional layer* akan menggunakan filter atau kernel pada setiap kumpulan vektor yang menjadi masukan. Kernel ini berupa array dua dimensi, bisa berukuran , , atau . Pada proses ini akan menghasilkan *feature map* yang menunjukkan fitur atau ciri dari kumpulan vektor inputan. Hasil *feature map* merupakan jumlah dari proses perkalian antara input data dengan kernel dengan indeks yang sama dan melakukan pergeseran kernel untuk nilai dari indeks berikutnya. Proses perkalian antara input data dengan kernel untuk menghasilkan *feature map* dapat dilihat pada Gambar 2.24.



**Gambar 2.24** *Convolution Layer* pada CNN (Yin, 2018)

Berdasarkan penjelasan tersebut dapat dirumuskan persamaan untuk menghasilkan *feature map* dengan inputan data , kernel dilambangkan dengan dan sebagai *feature map* yang dihasilkan pada Formula 2.10:

(2.10)

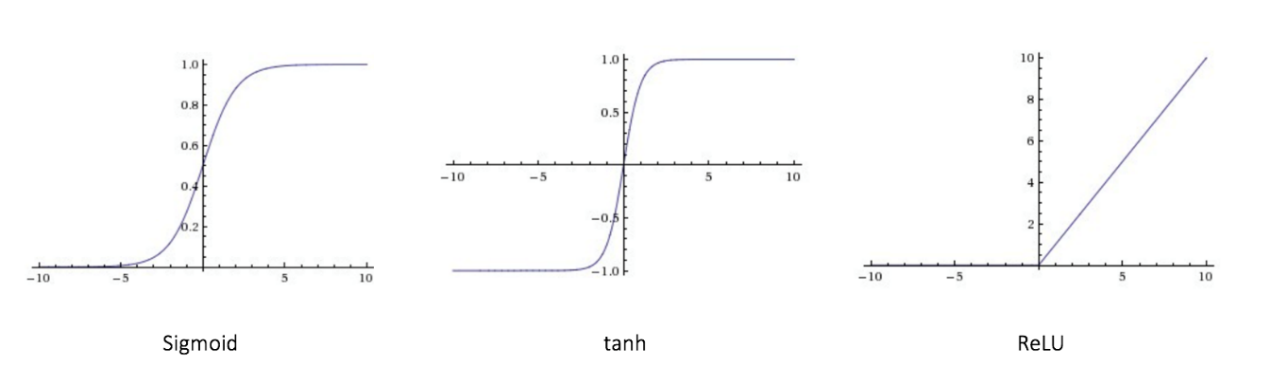
*Convolution layer* memiliki *hyperparameter* yang nilainya mempengaruhi *feature map* yang akan dihasilkan. *Hyperparameter* yang digunakan pada proses konvolusi yaitu *depth, stride* dan *zero-padding*. Penjelasan dari masing-masing *hyperparameter* dapat dilihat pada Tabel 2.1.

**Tabel 2.1** *Hyperparameter* pada *convolutional layer* (Zufar & Setiyono, 2016)

|  |  |  |
| --- | --- | --- |
| **No** | ***Hyperparameter*** | **Keterangan** |
| 1 | *Depth* | Kedalaman *layer* atau jumlah *layer* konvolusi |
| 2 | *Stride* | Jumlah pergeseran *filter* pada proses konvolusi |
| 3 | *Zero-padding* | Jumlah penambahan nilai intensitas nol di daerah sekitar input gambar |

* + 1. ***Activation Layer***

Tahap selanjutnya yaitu *activation layer*. Proses pada *activation layer* yaitu dengan memasukkan nilai *feature map* yang telah dihasilkan pada tahap sebelumnya ke dalam fungsi aktivasi. Fungsi aktivasi berfungsi untuk mengubah nilai pada *feature map* pada range tertentu dengan tujuan untuk meneruskan nilai yang menjadi fitur dominan ke layer selanjutnya. Beberapa fungsi aktivasi yang sering digunakan secara umum yaitu sigmoid, tanh dan ReLu: Fungsi aktivasi tersebut memiliki hasil grafik yang berbeda-beda, visualisasi grafik dari masing-masing fungsi aktivasi seperti pada Gambar 2.25



**Gambar 2.25** Contoh Grafik Fungsi Aktivasi (Tarabay, 2019)

Fungsi sigmoid terbentuk dari pembagian antara nilai 1 dengan penjumlahan nilai 1 dan exponensial pangkat minus . Persamaan fungsi sigmoid dapat dibentuk seperti Formula 2.11:

(2.11)

Fungsi selanjutnya yaitu Tanh. Persamaan dari fungsi ini adalah selisih eksponensial dari input dengan eksponensial minus nilai input dibandingkan dengan jumlah dari eksponensial nilai input dan eksponensial minus nilai input. Persamaan tersebut dapat ditulis seperti pada Formula 2.12:

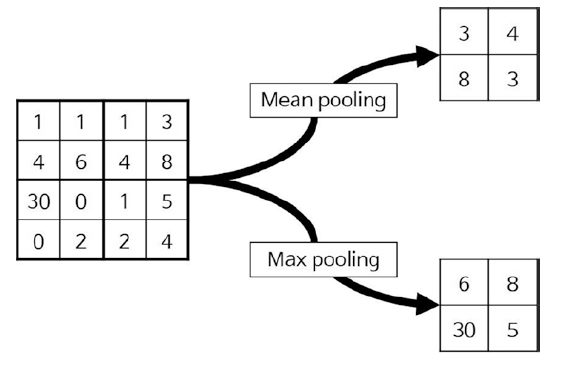
(2.12)

ReLU didefinisikan sebagai fungsi yang 2 kondisi untuk menghasikan nilainya. Jika nilai kurang dari 0 maka fungsi dari tersebut bernilai 0. Sedangkan apabila nilai lebih dari sama dengan 0 maka nilai dari fungsi adalah nilai tersebut. Persamaan dari fungsi ReLu dapat ditulis seperti pada Formula 2.13

(2.13)

* + 1. ***Pooling Layer***

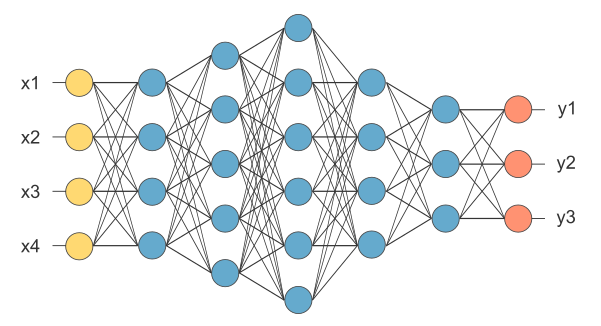
*Pooling layer* dapat disebut juga dengan *subsampling* atau *downsampling*. Hal tersebut karena *pooling layer* akan mengurangi dimensi dari *feature map* yang telah melalui *activation layer*. Pengurangan dimensi pada tahap ini tidak mempengaruhi informasi penting yang ada didalamnya. Proses *pooling* memiliki beberapa macam, seperti *maxpooling*, *meanpooling, sumpooling* dan sebagainya. Proses pada *pooling* dimulai dengan menentukan ukuran *downsampling,* misalnya . Setelah itu akan dilakukan proses *pooling* pada *feature map,* sebagai contoh *feature map* yang digunakan berukuran . Apabila menggunakan *maxpooling*, maka dari ukuran *downsampling* yang diambil nilai tertinggi untuk dijadikan nilai *feature map*. Sedangkan apabila menggunakan *meanpooling* maka nilai rata rata dari *downsampling* yang dijadikan nilai *feature map*. Proses *pooling* diilustrasikan seperti pada Gambar 2.26.



**Gambar 2.26** *Pooling Layer* (Tandungan, 2019)

* + 1. ***Fully Connected Layer***

*Fully connected layer* akan menggunakan hasil dari *pooling layer* sebagai inputan. Layer ini memiliki struktur yang sama dengan *neural network* pada umumnya yang terdiri dari *input layer*, *hidden layer* dan *output layer*. Masing-masing layer tersebut terdapat kumpulan *neuron* yang saling terhubungdengan *neuron-neuron* pada layer tetangganya. Berikut bentuk dari *full connected layer* dapat dilihat pada Gambar 2.27:



**Gambar 2.27** *Fully Connected Layer* (Tandungan, 2019)

Layer ini akan menentukan hasil klasifikasi berdasarkan data yang telah dimasukkan. Proses penentuan hasil klasifikasi menggunakan fungsi aktivasi . Fungsi aktivasi yaitu rasio eksponensial dari nilai input dengan jumlah nilai eksponensial. Persamaan dari fungsi aktivasi dapat dirumuskan seperti pada Formula 2.14:

(2.14)

dimana :

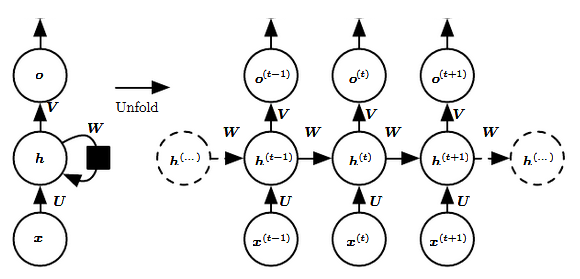
σ = probabilitas input tersebut merupakan class j

z = persamaan linear fungsi pre-aktivasi

j = salah satu class dari K class yang ada

* 1. ***Recurrent Neural Network* (RNN)**

*Recurrent Neural Network* atau RNN adalah algoritma *neural network* yang memiliki kemampuan *feedback* kembali ke *neuron* itu sendiri ataupun ke *neuron* lainnya sehingga aliran informasi dari *inputan* mempunyai arah jamak. Maka dari itu, RNN merupakan *neural network* yang digunakan untuk memproses data *sequence* atau data yang memiliki urutan dengan indeks dimulai dari data sampai data (Nabi, 2019). Penggunaan RNN pada data yang memiliki urutan data karena RNN memiliki memori sebagai pengingat histori masa lalu. RNN melakukan tugas yang sama untuk setiap elemen dari suatu urutan data, dimana *output* yang dihasilkan bergantung pada perhitungan sebelumnya. RNN memiliki minimal satu *feedback loop* di dalam *recurrent*. *Feedback loop* mempengaruhi kemampuan belajar dan kinerja jaringan (Udin et al., 2017). Proses dari algoritma RNN terdiri dari tiga bagian utama yaitu *forward pass, backward pass* dan *computing gradient*. Arsitektur dari RNN dapat diilustrasikan seperti pada Gambar 2.28.



**Gambar 2.28** Arsitektur *Recurrent Neural Network* (Goodfellow et al., 2016)

Berdasarkan arsitektur RNN diatas, dibagian sebelah kiri menunjukkan notasi dari RNN sedangkan bagian sebelah kanan merupakan jaringan yang belum terbuka namun jika terbuka akan masuk ke dalam jaringan penuh. Sebagai contoh, terdapat kalimat yang terdiri dari 5 kata maka jaringan dibuka menjadi 5 layer *neural network* dimana masing-masing layer mewakili satu kata. Dari arsitektur RNN terdapat beberapa atribut yang perlu diketahui:

1. *Input*  : merupakan data yang dimasukkan kedalam jaringan dengan panjang berupa vektor. Misalanya terdapat suatu kalimat yang memiliki 5 kata maka vektor dari kata pertama dari kalimat tersebut merupakan dan begitu pula seterusnya.
2. *Hidden state* () : merupakan jaringan tersembunyi dengan banyak dan akan bertindak sebagai memori dari jaringan. Nilai dari berdasarkan input terbaru dan hasil dari *hidden state* sebelumnya dengan rumus . Fungsi berdasarkan transformasi *non-linear* (*activation function*)yang digunakan, seperti tanh, sigmoid, ReLU dan lain sebagainya.
3. *Weights* () : Input yang akan dimasukkan ke dalam *hidden state* memiliki diparameterisasi bobot matriks , saat berada di *hidden state* ke *hidden state* selanjutnya diparameterisasi dengan bobot matriks , dan hasil dari *hidden state* ke *output* disebut bobot matriks . Semua bobot matriks () dibagi sepanjang waktu ().
4. *Output* () : menggambarkan hasil dari jaringan.
   * 1. ***Forward Pass***

Setelah membahas beberapa istilah dari RNN, dapat dilakukan *forward pass* untuk melakukan *training* modelnya. Misalnya terdapat beberapa konten berita yang terdiri dari beberapa kalimat dan berisi beberapa kata dengan *output* yang diharapkan berupa kelas dari beritanya. Untuk konsep *forward pass* akan ditinjau dari layer pada saat (berada ditengah). Karena RNN memperhatikan *output* dari data perhitungan sebelumnya pada , maka untuk menghitung *hidden state* dengan melakukan perkalian antara bobot matriks dengan paramter dan dijumlahkan dengan hasil perkalian antara parameter dan . Hasil tersebut lalu diproses menggunakan *activation function*, misalnya menggunakan tanh. Detail perhitungannya dapat dilihat pada Formula 2.12.

Setelah diproses dengan fungsi aktivasi dan mendapatkan hasilnya, maka nilai diteruskan ke *output* dengan melakukan fungsi . Bentuk persamaan dari fungsi aktivasi dapat dilihat pada Formula 2.14.

* + 1. ***Backward Pass***

*Backward pass* merupakan arsitektur RNN yang memperhatikan deret waktu sebelumnya. Apabila ingin mendapatkan nilai gradien pada waktu maka harus mengkalkulsi nilai gradien sampai dengan kondisi . Istilah ini sering disebut dengan *back-propagation through time* (BPPT) karena parameter dibagi oleh semua *time step* pada jaringan.

* + 1. ***Computing Gradients***

Dengan menggunakan *loss function ,* dapat digunakan untuk menghitung bobot dari matriks dengan istilah bias dan dan hasil pembaruannya *learning rate .* Untuk memperbaharui bobot untuk mengurangi *loss* dengan menghitung selisih antara bobot terakhir dengan hasil perkalian antara nilai alpha dengan rasio partial dan . Persamaan dari memperbaharui bobot dapat dibuat formula seperti pada Formula 2.15.:

(2.15)

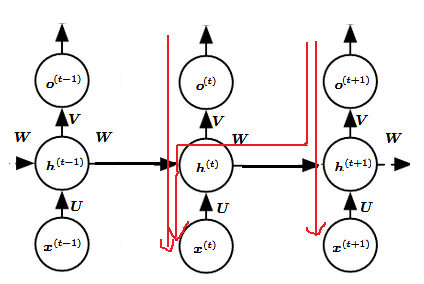
Hal yang sama juga dilakukan pada bobot dan .

Selanjutnya menghitung gradien dengan menggunakan konsep BPPT pada RNN. Berdasarkan grafik komputasi terdapat parameter dan dengan urutan data dan memiliki indeks tertentu terhadap . Dari setiap *node*  perlu menghitung gradien secara rekursif.

Gradien untuk *output* merupakan hasil kalkulasi dari yang telah digunakan *activation function* untuk mendapatkan probabilitas dari vektor . Dengan asumsi bahwa *loss* merupakan kemungkinan negatif dari target sebelumnya. Persamaan tersebut dapat ditulis seperti pada Formula 2.16.

(2.16)

Setelah mengetahui bagaimana gradien mengalir melalui *hidden state,* sekarang dapat dilihat dengan jelas melihat diagram pada waktu mengalir gradien dari *output* saat ini ke *hidden state* selajutnya. Proses aliran nilai gradien dapat diperjelas pada Gambar 2.29.

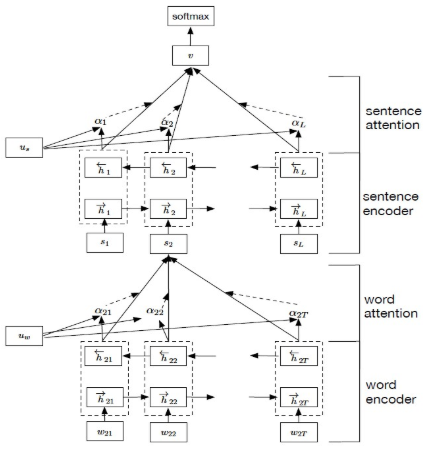


**Gambar 2.29** Alur Gradien pada RNN (Nabi, 2019)

Berdasarkan semua penjelasan mengenai RNN diatas, dapat disimpulkan beberapa tahapan yang dilakukan untuk menggunakan ataupun menerapkan algoritma RNN dalam suatu kasus yang ingin diselesaikan. Tahapan-tahapannya sebagai berikut:

1. Melakukan inisialisasi bobot matriks dengan distribusi acak dan bias dengan nol.
2. Melakukan *forward propagation* untuk menghitung nilai prediksi
3. Hitung nilai *loss*
4. Lakukan proses *back-propagation* untuk menghitung gradien
5. Perbaharui nilai bobot berdasarkan gradien
6. Ulangi langkah 2-5
   1. ***Hierarchical Attention Network* (HAN)**

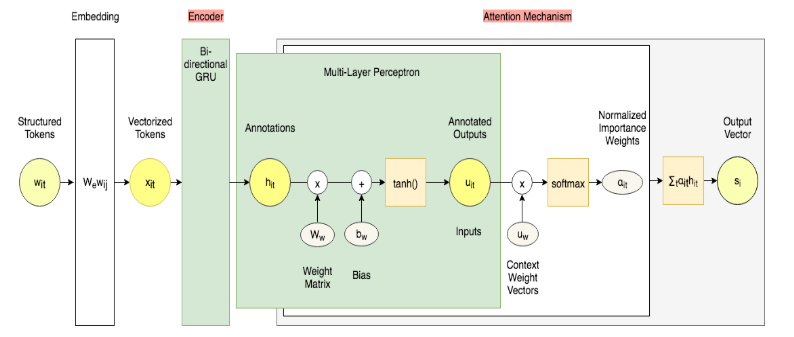
*Hierarchical attention network* (HAN) merupakan algoritma *neural network* yang mempertimbangan struktur hierarki suatu dokumen. Hierarki tersebut merepresentasikan gabungan antara representasi dokumen dengan representasi kalimat. Representasi kalimat dibangun terlebih dahulu dengan *encode* kata dari suatu kalimat dan melalui *attention mechanism* untuk menghasilkan vektor yang merepresentasikan suatu kalimat. Representasi dokumen juga menggunakan cara yang sama, bedanya yang menjadi inputannya adalah vektor kalimat. HAN memiliki arsitektur yang terdiri dari *word level, word encoder, word attention,* *sentence level, sentence encoder* dan *sentence attention*. Berikut arsitektur dari algoritma HAN seperti pada Gambar 2.30.:



**Gambar 2.30** Arsitektur *Hierarchical Attention Network* (Kränkel & Lee, 2018)

* + 1. ***Word Level***

Tahap ini adalah tahap awal pada algoritma HAN. Prosesnya akan dimulai dari data hasil *preprocessing* yang berupa kumpulan token yang terstruktur yang akan menjadi inputan pada tahap ini. Sedangkan *output* yang akan dihasilkan setelah melalui tahapan ini berupa vektor untuk masing-masing token. Berikut alur dari tahap *word level* yang terdiri dari *word encoder* dan *word attention* diilustrasikan pada Gambar 2.31:



**Gambar 2.31** Alur pada tahap *word layer* (Kränkel & Lee, 2018)

Token yang terstruktur merupakan representasi dari kata ke dari kalimat . Karena model tidak dapat memproses data berupa teks biasa atau tipe data string, maka diperlukan proses *embedding*. Ada beberapa algoritma yang populer yang dapat digunakan untuk melakukan proses *embedding* yaitu word2vec dan GloVe. Layer *embedding* akan menghasilkan vektor multidimensional untuk masing-masing token . Dengan cara tersebut, token-token akan direpresentasikan secara numerik sebagai . Dari penjelasan tersebut dapat dirumuskan seperti pada Formula 2.17:

(2.17)

* + 1. ***Word Encoder***

Setelah mendapatkan vektor dari masing-masing token, tahap berikutnya dengan melakukan proses *encoding*. Proses *encoding* akan menggunakan *Gated Recurrent Network* (GRU). GRU akan memprediksi suatu kata bergantung pada kata sebelumnya. GRU memiliki *hidden state* sebagai sel memori untuk transfer informasi. Terdapat dua gerbang yang memutuskan apakah suatu informasi disimpan atau dilupakan sekaligus melakukan pembaharuan informasi pada model. Tujuan dari layer ini akan mengekstrak data yang revelan dari suatu kalimat. Hal ini biasa disebut dengan *annotations*. Pada model ini, *bidirectional* GRU yang diterapkan untuk mendapatkan *annotation* kata dengan meringkas dari dua arah dengan variabel . Persamaan yang akan terbentuk menjadi Formula 2.18, 2.19 dan 2.20.

(2.18)

(2.19)

(2.20)

* + 1. ***Word Attention***

*Annotation* akan membangun *attention* yang dimulai dari *hidden layer* lainnya. Tujuannya untuk membiarkan model belajar dengan bobot dan bias yang diinisialisasi secara acak. *Annotation* akan terus diperbaharui dan direpresentasikan oleh . Selanjutnya, proses *actiovation function* dari hasil *annotation* dengan menggunakan fungsi . Fungsi ini akan mengoreksi nilai input antara -1 dan 1 serta memetakan nol hingga mendekati nol. Persamaan tersebut dapat ditulis seperti pada Formula 2.21.

(2.21)

Anotasi baru akan dikalikan dengan vektor yang telah dilatih . Setelah itu, akan dinormalisasi untuk mendapatkan bobot penting per kata menggunakan fungsi . Persamaan tersebut dapat ditulis seperti pada Formula 2.22.

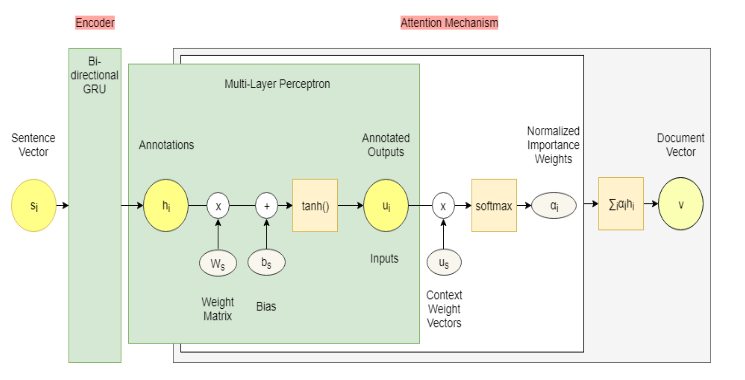
(2.22)

Bobot-bobot penting yang telah didapatkan disatukan dan dijumlah seluruhnya. Hasil dari penjumlah semua bobot disebut vektor kalimat. Rumus untuk mendapatkan vektor kalimat seperti pada Formula 2.23.

(2.23)

* + 1. ***Sentence Level***

Selanjutnya jaringan akan dijalankan pada level kalimat dengan prosedur yang sama seperti level kata, tetapi sekarang fokus pada kalimat . Bedanya, pada level kata terdapat *embedding layer* sedangkan pada level kalimat tidak perlu *embedding layer*. Hal tersebut dikarenakan inputan pada level kalimat sudah berupa vektor kalimat yang merupakan hasil dari level kata. Berikut jalannya proses pada *sentence level* dapat dilihat pada Gambar 2.31.



**Gambar 2.31** Alur pada tahap *sentence level* (Kränkel & Lee, 2018)

* + 1. ***Sentence Encoder***

Konteks kalimat akan dirangkum menggunakan *bidirectional* GRU dengan alur maju dan mundur. Persamaannya dapat dirumuskan seperti pada Formula 2.24, 2.25, dan 2.26.

(2.24)

(2.25)

(2.26)

* + 1. ***Sentence Attention***

Bobot dan bias yang didapat akan dilatih kembali dengan inisialisasi secara acak. Hasil akhirnya yaitu berupa vektor dokumen yang bisa digunakan sebagai fitur untuk melakukan klasifikasi dokumen. Persamaan tersebut dapat ditulis seperti pada Formula 2.27, 2.28, dan 2.29.

(2.27)

(2.28)

(2.29)

* 1. **Penelitian Terdahulu**

Penelitian yang berkaitan dengan *hoax* pernah dilakukan menggunakan metode *naïve bayes* dengan tingkat akurasi 80% untuk mengklasifikasikan 3 kategori yaitu berita palsu (*hoax*), berita asli (*fact*) dan berita yang tidak berindikasi (Tanjung, 2018). Selain itu, ada penelitian lain yang membandingkan antara algoritma *rocchio* dengan *multinomial naïve bayes* dengan akurasi yang didapatkan sebesar 83,501% untuk *rocchio* dan 65,835% untuk *multinomial naïve bayes* (Afriza & Adisantoso, 2018). Algoritma *support vector machine* pun pernah digunakan untuk klasifikasi artikel *hoax* dengan tingkat akurasi yang didapatkan sebesar 95,8333% (Maulina & Sagara, 2018). Ada beberapa metode lain yang pernah digunakan seperti *gradient descent backpropagation* dengan tingkat akurasi 72,04% sedangan *lavenberg-marquad backpropagation* memiliki akurasi 72,19% dalam klasifikasi pengguna media sosial (Lhaksmana et al., 2017). Penerapan *deep learning* juga pernah dilakukan dalam klasifikasi sentimen analisis *tweet* dengan menggunakan metode *deep belief network* dengan tingkat akurasi sebesar 93,31% (Zulfa & Winarko, 2017). Algoritma *convolutional neural network* juga pernah digunakan pada penelitian tentang klasifikasi artikel berita berbahasa indonesia dengan nilai akurasi sebesar 96,70% dan presisi, recall serta f-measure mencapai 96,60%. Berdasarkan penelitian terdahulu yang telah dipaparkan, dapat dirangkum seperti pada Tabel 2.2

**Tabel 2.2** Penelitian Terdahulu

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Penulis** | **Judul** | **Metode** | **Keterangan** |
| 1 | Budi Setiawan Tanjung.  2018 | Pendekatan *Text Mining* sebagai Sistem Pendeteksi Pemberitaan Palsu yang Tersebar dalam Twitter | *Naive Bayes Classifier* | Pola berita palsu belum dapat dikenali secara jelas karena banyak berita yang hampir sama. |
| 2 | Aulia Afriza dan Julio Adisantoso.  2018 | Metode Klasifikasi *Rocchio* untuk Analisis *Hoax* | *Rocchio* dan Multinomial *Naive Bayes* | Pada penelitian ini tidak memiliki fitur untuk menambahkan model pada data latih |
| 3 | Kemas Muslim Lhaksmana, Fhira Nhita dan Ageng Budhiarto.  2017 | Klasifikasi Pengguna Media Sosial Twitter dalam Persebaran *Hoax* Menggunakan Metode *Backpropagation* | *Gradient Descent Backpropagation* dan *Lavenberg Marquad Backpropagation* | Performa sistem baik, namun mendapatkan error yang cukup besar karena adanya kemiripan karakteristik pada data dan membutuhkan waktu pelatihan yang relatif lama |
| 4 | Dina Maulina dan Rofie Sagara.  2018 | Klasifikasi Artikel *Hoax* Menggunakan *Support Vector Machine Linear* dengan Pembobotan *Term Frequency-Inverse Document Frequency* | *Support Vector Machine* | Data yang digunakan tergolong sedikit (108 artikel *hoax* dan 132 artikel non *hoax*) |
| 5 | Ira Zulfa dan Edi Winarko.  2017 | Sentimen Analisis Tweet Berbahasa Indonesia dengan *Deep Belief Network* | *Deep Belief Network* | Penggunaan metode *Deep Belief Network* dengan *BagofWord* sebagai fitur ekstraksi terbukti tidak memberikan akurasi lebih baik |
| 6 | Ar Razi.  2017 | Klasifikasi Artikel Berita Berbahasa Indonesia Menggunakan *Convolutional Neural Network* | *Convolutional Neural Network* | Jumlah dataset pada proses *training* masih sedikit dan pada *preprocessing* tidak menggunakan *stemming* serta *stopword* sehingga hasilnya kurang optimal |

**BAB III**

**METODOLOGI PENELITIAN DAN PENGEMBANGAN SISTEM**

**3.1 Metodologi Penelitian**

Pada bagian ini akan dibahas mengenai metodologi penelitian yang digunakan dan analisis kebutuhan sistem dalam penelitian tentang Komparasi Metode *Deep Learning* Untuk Klasifikasi Berita *Hoax* Menggunakan *Convolutional Neural Network, Recurrent Neural Network* dan *Hierarchical Attention Network*.

Metodologi penelitian yang digunakan pada penelitian ini bermodel kualitatif. Penggunaan model tersebut karena penelitian ini bersifat deskriptif dan cenderung menggunakan analisis. Selain itu, penelitian ini berdasarkan data, memanfaatkan teori yang sudah ada menjadi bahan penjelas dan penguat argumen serta berakhir dengan suatu teori. Metode penelitian yang dilakukan meliputi pengumpulan data, analisis kebutuhan sistem, perancangan sistem, penulisan program dan pengujian. Metode penelitian yang dilakukan dapat dilihat pada Gambar 3.1.

3.2

**Daftar Pustaka**

Abdullah, A. H. (2018). Word Embedding. Retrieved February 29, 2020, from https://rpubs.com/ website: https://rpubs.com/ahmadhusain/wordembedding

Afriza, A., & Adisantoso, J. (2018). Metode Klasifikasi Rocchio untuk Analisis Hoax. *Jurnal Ilmu Komputer Dan Agri-Informatika*, *5*(1), 1–10. https://doi.org/10.29244/jika.5.1.1-10

Assidik, G. K. (2018). Kajian Identifikasi dan Upaya Penangkalan Pemberitaan Palsu (Hoax) Pada Pembelajaran Bahasa Indonesia. *Kongres Bahasa Indonesia*.

Berry, M. W., & Kogan, J. (2010). *Text Mining : Applications and Theory*. United Kingdom: WILEY.

Fajri, D. (2019). Menkominfo: Ada 900 Ribu Situs Penyebar Informasi Hoax. Retrieved June 29, 2019, from www.okezone.com website: https://news.okezone.com/read/2019/02/14/337/2018062/menkominfo-ada-900-ribu-situs-penyebar-informasi-hoax

Feldman, R., & Sanger, J. (2007). *The Text Mining Handbook : Advanced Approaches in Analyzing Unstructured Data* (1st ed.). New York: Cambridge University Press.

Gaikwad, S. V., Chaugule, A., & Patil, P. (2014). Text Mining Methods and Techniques. *International Journal of Computer Applications*, *85*(17), 42–45. https://doi.org/10.5120/14937-3507

Goller. (2000). Automatic Document Classification: A Thorough Evaluation of Various Methods. *Proceedings of International Symposium on Information Theory and Its Application*, 145–162.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.

Haddi, E., Liu, X., & Shi, Y. (2013). The Role of Text Pre-processing in Sentiment Analysis. *Procedia Computer Science*, (17), 26–32.

Hashimi, H., Hafez, A., & Mathkour, H. (2015). Selection criteria for text mining approaches. *College of Computer and Information Sciences*.

Kotler, P., & Keller, K. L. (2012). *Marketing Management* (14th ed.). United State of America: Pearson.

Kränkel, M., & Lee, H.-E. (2018). Text Classification with Hierarchical Attention Networks. Retrieved February 11, 2020, from Seminar Information System website: https://humboldt-wi.github.io/blog/research/information\_systems\_1819/group5\_han/

Lhaksmana, K. M., Nhita, F., & Budhiarto, A. (2017). Klasifikasi Pengguna Media Sosial Twitter Dalam Persebaran Hoax Menggunakan Metode Backpropagation Classification of Users Social Media Twitter in the Hoax Spread. *E-Proceeding of Engineering*, *4*(2), 3082–3090.

Librianty, A. (2017). Survei: Media Sosial Jadi Sumber Utama Penyebaran Hoax.

MacDougall, C. D. (1958). *Hoaxes*. New York: Dover Publication.

Maheshwari, A. (2018). *Report on Text Classifiction using CNN, RNN & HAN*.

Maulina, D., & Sagara, R. (2018). Klasifikasi Artikel Hoax Menggunakan Support Vector Machine Linear Dengan Pembobotan Term Frequency – Inverse Document Frequency. *Jurnal Mantik Penusa*, *2*(1), 35–40.

Nabi, J. (2019). Recurrent Neural Networks (RNNs). Retrieved February 11, 2020, from https://towardsdatascience.com/ website: https://towardsdatascience.com/recurrent-neural-networks-rnns-3f06d7653a85

Nasution, M. A. (2017). HOAX SEBAGAI BENTUK HUDUD MENURUT HUKUM ISLAM. *Yurisprudentia*, *3*(1).

Nazief, B., & Adriani, M. (1996). *Confix Stripping: Approach to Stemming Algorithm for Bahasa Indonesia*.

Nikhit, K. (2015). A Web Scraping Approach in Node.js. *International Journal of Science, Engineering and Technology Research*, 909–912.

Pereira, R. C., & Vanitha, T. (2015). Web Scraping of Social Networks. *International Journal of Innovative Research in Computer and Communication Engineering*, *3*(7), 237–240.

Pressman, R. S. (2010). *Software Engineering : A Practitioner’s Approach* (7th ed.). McGraw-Hill Education.

Primas, N. (2017). *Deteksi Cacat Permukaan Buah Manggis Berbasis Pengolahan Citra Menggunakan Metode Klasifikasi Support Vector Machine (SVM)*. Universitas Muhammadiyah Yogyakarta.

Razi, A. (2017). *Klasifikasi Artikel Berita Berbahasa Indonesia Menggunakan Convolutional Neural Network*. Universitas Gadjah Mada.

Respati, S. (2017). Mengapa Banyak Orang Mudah Percaya Berita “Hoax”? Retrieved February 9, 2020, from www.kompas.com website: https://nasional.kompas.com/read/2017/01/23/18181951/mengapa.banyak.orang.mudah.percaya.berita

Safko, L. (2012). *The Social Media Bible : Tactics, Tools and Strategies for Business Success* (3rd Editio). New York: John Wiley & Sonic Inc.

Sebastiani, F. (2002). Machine Learning in Automated Text Categorization. *ACM Computing Surveys*, *34*(1), 1–47.

Tandungan, S. (2019). Pengenalan Convolutional Neural Network – Part 1. Retrieved February 13, 2020, from http://sofyantandungan.com/ website: http://sofyantandungan.com/pengenalan-convolutional-neural-network-part-1/

Tanjung, B. S. (2018). Pendekatan Text Mining sebagai Sistem Pendeteksi Pemberitaan Palsu yang Tersebar dalam Twitter. *Universitas Widya Kartika*, 1–6.

Tarabay, M. R. (2019). PyTorch and deep learning. Retrieved February 13, 2020, from https://rafietarabay.blogspot.com/2019/05/?m=0

Turland, M. (2010). *php|architect’s Guide to Web Scraping with PHP*. Los Angeles.

Udin, M., Kaloko, B. S., Hardianto, T., Elektro, J. T., Teknik, F., Unej, U. J., & Kalimantan, J. (2017). Peramalan Kapasitas Baterai Lead Acid pada Mobil Listrik Berbasis Levenberg Marquardt Neural Network. *BERKALA SAINSTEK*, *2*, 112–117.

Vibriza, J., Rahadi, D. R., Marwan, M. R., & Ahyad. (2017). Perilaku pengguna dan informasi. *Jurnal Gunadarma*, *4*(1), 192–208. https://doi.org/https://doi.org/10.22146/jps.v4i2.28586

Wearesocial. (2019). Digital 2019: Indonesia. Retrieved June 29, 2019, from www.datareportal.com website: https://datareportal.com/reports/digital-2019-indonesia/

Wijayanto, F. I. (2012). Social Media: Definisi, Fungsi, Karakteristik. Retrieved February 9, 2020, from www.prezi.com website: https://prezi.com/vddmcub\_-ss\_/social-media-definisi-fungsi-karakteristik/

Wismadi, I. M. (2019). *Deteksi Kematangan Buah Naga Siap Panen dengan Pendekatan Deep Learning Menggunakan Metode Smaller VGGNet-like Network*. Universitas Udayana.

Xing, W., & Du, D. (2018). Dropout Prediction in MOOCs : Using Deep Learning for Personalized Intervention. *Journal of Educational Computing Research*, (March). https://doi.org/10.1177/0735633118757015

Yin, L. (2018). A Summary of Neural Network Layers. Retrieved February 13, 2020, from https://medium.com/ website: https://medium.com/machine-learning-for-li/different-convolutional-layers-43dc146f4d0e

Zhang, Y., & Wallace, B. (2015). *A Sensitivity Analysis of (and Practitioners’ Guide to) Convolutional Neural Networks for Sentence Classification*.

Zufar, M., & Setiyono, B. (2016). Convolutional Neural Network untuk Pengenalan Wajah Secara Real-Time. *Jurnal Sains Dan Seni ITS*, *5*(2), 72–77.

Zulfa, I., & Winarko, E. (2017). Sentimen Analisis Tweet Berbahasa Indonesia Dengan Deep Belief Network. *IJCCS (Indonesian Journal of Computing and Cybernetics Systems)*, *11*(2), 187–198. https://doi.org/10.22146/ijccs.24716