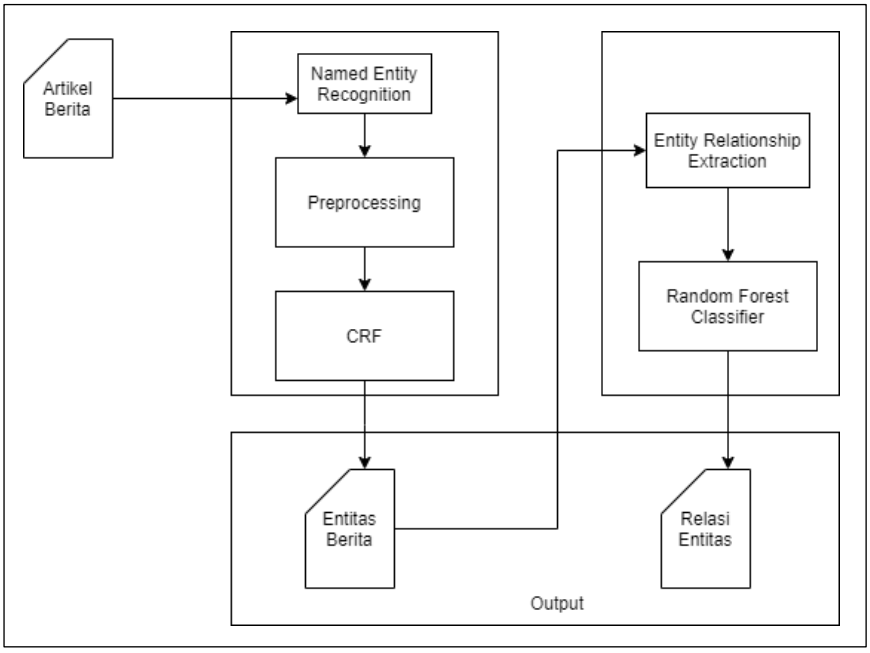
BAB II

LANDASAN TEORI

2.1 Named Entity Recognition.

Named Entity Recognition (NER) berfungsi untuk mengklasifikasikan setiap kata pada suatu dokumen kedalam kategori yang telah dibuat sebelumnya. NER termasuk kedalam *information extraction* yang berfungsi untuk meng-ekstrak informasi tertentu dari sebuah dokumen. (Zhou dan Su, 2001) NERpada penelitian akan dipakai untuk mendeteksi Entitas yang berada pada artikel berita Bahasa Indonesia. NER akan dipakai untuk mendeteksi keberadaan entitas seperti nama seseorang, nama organisasi dan juga nama lokasi. Pada penelitian ini disediakan *pipeline* yang menjelaskan bagaimana *entity relation* akan di ekstrak dari sebuah artikel berita.



Gambar 2.1 Pipeline Named Entity Recognition (Magge dkk., 2018)

2.2 Data Preprocessing

Proses persiapan data pada dasarnya berakar kepada NLP. *Text preprocessing* mengambil masukan data mentah teks dan memprosesnya menjadi Token yang bisa dipakai. Token merupakan suatu kata atau kelompok kata yang didapat dari betapa seringnya kata tersebut muncul yang kemudian dipakai sebagai fitur untuk dianalisa (Anandarajan dkk., 2019)

2.2.1 Stop Word Removal

Proses ini merupakan proses di mana kata-kata pengisi atau kata penghenti akan dihilangkan. Kata-kata yang sering muncul seperti, dan, dengan, atau, ini, tetapi, adalah, yaitu, dan lain-lain. Kata-kata tersebut dihilangkan karena pada penelitian ini kata-kata di atas hanya berperan sebagai kata penghenti ataupun penyambung untuk membuat suatu kalimat menjadi lebih bermakna dan lebih gampang dipahami.

Tabel 2.1 Contoh *stopword removal* (Anandarajan dkk., 2019)

Sebelum *stopword Removal*	[Budi] [membeli] [baju] [dan] [juga] [topi]
Setelah *stopword Removal*	[Budi] [membeli] [baju] [topi]

2.2.2 Tokenization

Proses ini berfungsi untuk memisahkan suatu kata-kata yang mendirikan suatu kalimat menjadi kata individu ataupun token. Contoh : “Rex dan

Cookie merupakan kedua anjing kesayangan saya" pada kalimat contoh tersebut dapat diproses menjadi 9 token.

Tabel 2.2 Contoh *tokenization* (Anandarajan dkk., 2019)

Kalimat Biasa	Rex dan Cookie merupakan kedua anjing kesayangan saya.
Hasil Tokenization	[Rex] [dan] [Cookie] [merupakan] [kedua] [anjing] [kesayangan] [saya] [.]

2.2.3 Stemming

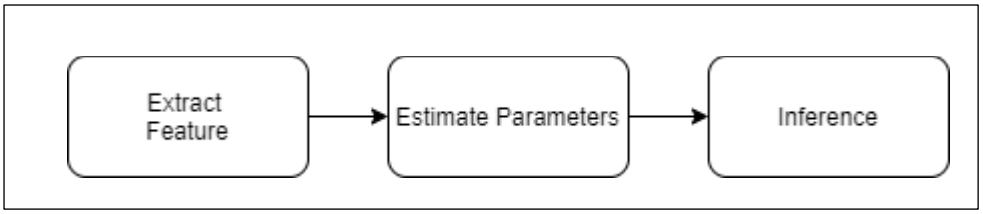
Proses yang berfungsi untuk memecahkan sebuah kata menjadi kata yang paling dasar. Stemming berfungsi dalam penghilangan *suffix* atau *preffix* dari suatu kata untuk mengurangi ukuran dari kata tersebut. (Anandarajan dkk., 2019). Contoh yang bisa kita ambil adalah kata "Melatih" dan "Latihan" setelah proses stemming dilakukan kedua kata tersebut akan menghasilkan kata latih yang merupakan kata dasar dari kedua kata tersebut.

Tabel 2.3 Contoh stemming (Anandarajan dkk., 2019)

Sebelum *stemming*	[Budi] [memakan] [gado-gado] [memakai] [sumpit]
Sesudah stemming	[Budi] [makan] [gado-gado] [pakai] [sumpit]

2.3 Conditional Random Fields.

Conditional Random Fields (CRF) merupakan suatu model probablistik yang banyak digunakan pada proses segmentasi dan pelabelan suatu sekuen data. CRF merupakan metode percampuran antara *Hidden Markov Model* (HMM) dan *Maximum Entropy Markov Model* (MEMM) (Akbar dkk., 2015) CRF pada lapisan output telah ditemukan sangat efisien untuk berbagai jenis *sequence tagging* (Reimers dan Gurevych, 2017).



Gambar 2.1 Proses inti pada CRF(Akbar dkk., 2015)

Extract Feature merupakan proses paling penting dalam pembuatan model. Pada proses ini dibuat *feature function* berdasarkan nilai-nilai fitur yang ada pada setiap kalimat pada data latih (Akbar dkk., 2015). Fitur ini juga berfungsi untuk mengkombinasikan sebuah fitur dengan labelnya yang bersesuai antara label yang berdekatan. *Estimate parameter* merupakan tahap mendapatkan nilai optimal untuk parameter pendukung *feature function*. Nilai optimal parameter fungsi dapat dihitung dengan prosedur maksimum *likelihood* (Akbar dkk., 2015). Prosedur ini berfungsi untuk memaksimalkan kemiripan ciri dari data pelatihan yang telah dimodelkan dari Conditional Random Fields.

Conditional Random Fields (CRF) pertama kali diperkenalkan oleh Lafferty sebagai suatu model sequence data labeling yang berbasis pendekatan statistik, CRF

telah menunjukan tingkat kesuksesan yang empiris pada penelitian NER karena CRF terlepas dari permasalahan label bias dengan menggunakan normalisasi global (Li dkk., 2015).

1. Hasil dari NER merupakan suatu sequence label sehingga kita akan menggunakan *linear-chain* CRF.

$$P(y|x) = \frac{1}{Z(x)} \exp\left(\sum_{K=1}^{K} \lambda_k f_k(y_t, y_{t-1}, x_t)\right) \quad \ldots(2.1)$$

   Definisikan y dan x merupakan vector acak dan $f_k(y_t, y_{t-1}, x_t)$ merupakan *feature function* (k merupakan angka), $\lambda_k$adalah *learned weight* dari fitur, y_t, y_{t-1} secara berurutan bertuju kepada kondisi sebelum dan kondisi sekarang,.

2. $Z(x)$ adalah faktor normalisasi dengan *state sequence* sebagai berikut :

$$Z(x) = \sum_{y} exp\left(\sum_{K=1}^{K} \lambda_k f_k(y_t, y_{t-1}, x_t)\right) \quad \ldots(2.2)$$

   Masalah utama pada CRF algoritma *linear-chain* adalah bagaimana cara untuk mencari parameter vector $\vec{\lambda} = \{\lambda_1, \ldots, \lambda_k\}$.cara ini dilakukan ketika proses *training*. Proses *training* dilakukan untuk mencari parameter estimasi dan metode yang paling sering dipakai untuk mencari parameter estimasi adalah menggunakan metode *maximum likelihood*.

3. Anggap setiap $(x^i|y^i) \in D = \{(x^i|y^i)\}_{i=1}^{N}$ tersebar secara independen dan identik, dimana setiap $x^i = \{x_1^i, x_2^i, \dots, x_t^i\}$ adalah sekuens *input* dan setiap $y^i = \{y_1^i, y_y^i, \dots, y_t^i\}$ adalah sekuens dari hasil prediksi yang bersangkutan. oleh karena itu fungsi *log-likelihood* dari data *training D* akan ditampilkan pada persamaan 2.3

$$L(\vec{\lambda}) = \sum_{i=1}^{n} \log P(y^i|x^i). \quad \dots(2.3)$$

4. Dari persamaan (1) dan persamaan (3), fungsi log-likelihood akan seperti ini:

$$L(\vec{\lambda}) = \sum_{i=1}^{N} log\left[\frac{1}{Z(x^i)} \exp\left(\sum_{t=1}^{T}\sum_{i=1}^{K} \lambda_k f_k(y_t^i, y_{t-1}^i, x_t^i)\right)\right] \quad \dots(2.4)$$

$$= \sum_{i=1}^{N}\sum_{t=1}^{T}\sum_{k=1}^{K} \lambda_k f_k(y_t^i, y_{t-1}^i, x_t^i) - \sum_{i=1}^{N} \log Z(x^i)$$

5. Untuk menghindari *overfitting*, maka persamaan (4) akan menjadi :

$$\boldsymbol{L(\vec{\lambda}) = L(\vec{\lambda}) - \sum_{K=1}^{K} \frac{\lambda_k^2}{2\sigma^2}.} \quad \dots(2.5)$$

6. Dari persamaan (4) dan (5) maka fungsi *log-likelihood* akan menjadi:

$$L(\vec{\lambda}) = \sum_{i=1}^{N}\sum_{t=1}^{T}\sum_{K=1}^{K} \lambda_k f_k(y_t^i, y_{t-1}^i, x_t^i) - \sum_{i=1}^{N} \log Z(x^i) - \sum_{k=1}^{K} \frac{\lambda_k^2}{2\sigma^2}. \quad \dots(2.6)$$

Rumus perhitungan CRF diatas dengan menggunakan metode *log-likelihood* diangkat langsung dari penelitian sebelumnya yang terdapat pada jurnal (Li dkk., 2015).

2.4 Random Forest Classifier

Random Forest Classifier merupakan sebuah koleksi ataupun kumpulan dari pohon klasifikasi dan regresi dilatih pada dataset yang memiliki ukuran yang sama dengan ukuran *training set*, yang disebut dengan nama *bootstrap*, setelah pohon terbuat, kumpulan *bootstrap* yang tidak berisikan data set original akan dipakai sebagai data yang akan di uji coba. (Sarica dkk., 2017).

Model dari Random Forest berbasis pada K *decision trees.* Setiap pohon akan memilih satu pohon dimana variable x paling sesuai dengan pohon tersebut. dan hanya akan ada satu suara yang diberikan kepada pohon tersebut (Tan dkk., 2019).

1. K merupakan jumlah dari *decision trees* yang berada di dalam suatu *random forest*. θ_k merepresentasikan *random vectors* yang independen dan terbagi secara identik. Persamaan *k decision trees* dapat didefinisikan pada persamaan 2.7.

$$\{h(X, \theta_k), k = 1\ , 2\ , \dots , k \qquad \dots(2.7)$$

2. Fitur yang sangat penting dari Random Forest untuk mencerminkan *generalization performance* yaitu kesamaan dan hubungan dari *decision trees*. *Generalization* merupakan kemampuan sistem untuk menentukan keputusan-keputusan yang tepat pada data baru dengan distribusi di luar

data *training*. *Generalization error* mencerminkan kemampuan *generalization* dari sistem. *Generalization* yang lebih kecil akan menghasilkan performa yang lebih baik. Persamaan *generalization error* dapat didefinisikan pada persamaan 2.8.

$$PE^* = P_{X,Y}(mr(X,Y) < 0) \quad \ldots(2.8)$$

PE^* merepresentasikan *generalization error*, X,Y mengindikasikan ruang probabilitas dan $mr(X,Y)$ merupakan *margin function*.

3. *Margin function* merepresentasikan berapa banyak suara yang benar dalam mengklasifikasikan suatu sampel X dibandingkan dengan banyak suara yang salah mengklasifikasikan sampel tersebut, semakin besar nilai dari sebuah *margin function* maka semakin besar *confidence* dari *classifier* tersebut. Persamaan *margin function* didefinisikan pada persamaan 9

$$mr(X,Y) = avg_k I(h(X,\theta_k) = Y) - max_{j \neq y} avg_k I(h(X,\theta_k) = J) \quad \ldots(2.9)$$

Y merepresentasikan klasifikasi yang benar dan J merepresentasikan klasifikasi yang salah. $I(g)$ merupakan *indicative function*, $avg_k(g)$ merepresentasikan *means averaging* dan $h(g)$ merepresentasikan *sequence of classification model*.

4. *Covergence expression* dari *generalization error* dijelaskan pada persamaan 2.10.

$$\lim_{k \to \infty} PE^* = P_{X,Y}(P_\theta\left(I(h(X,\theta_k) = Y)\right) - max_{J \neq Y} P_\theta(I(h(X,\theta_k) = J))) \quad \ldots(2.10)$$

Persamaan 2.10 mengindikasikan bahwa *generalization error* akan lebih sering terpaku kepada batas atas sehingga model ini tidak akan *overfiting* dengan bertambahnya jumlah *decision trees*. Model Random Forest bertujuan untuk menciptakan sebuah random forest dengan korelasi rendah namun dengan *classification intensity* yang tinggi.

5. *Classification intensity* merupakan ekspektasi matematikal dari $mr(X,Y)$ pada keseluruhan ruang sampel. *Classification intensity* dapat didefinisikan dengan persamaan 11.

$$S = E_{X,Y} mr(X,Y) \quad \ldots(2.11)$$

6. θ dan θ^I adalah independen dan merupakan *identically distributed vector* dan merupakan *correlation coefficients* dari $mr(\theta,X,Y)$ dan $mr(\theta^I,X,Y)$ hal ini didefinisikan pada persamaan 12.

$$\bar{\rho} = \frac{cov_{X,Y}(mr(\theta,X,Y), mr(\theta^I,X,Y))}{sd(\theta)sd(\theta^I)} \quad \ldots(2.12)$$

Persamaan 12 mendefinisikan bahwa korelasi diantara pohon-pohon $h(X,\theta)$ dan $h(X,\theta^I)$ pada dataset X dan Y dapat dihitung dengan $\bar{\rho}$, semakin besarnya $\bar{\rho}$ semakin besarlah *correlation coefficient.*

7. $sd(\theta)$ pada persamaan 12 dapat diekspresikan menjadi persamaan 13

$$sd(\theta) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(mr(x_i,\theta) - \frac{1}{N}\sum_{i=1}^{N}(mr(x_i,\theta))^2} \quad \ldots(2.13)$$

8. Persamaan 14 mendefinisikan batas atas ataupun *upper bound* dari *generalization error.*

$$P_{X,Y}(mr(X,Y) < 0) \leq \frac{\bar{\rho}(1 - S^2)}{S^2} \quad \ldots(2.14)$$

Dapat dilihat bahwa *generalization error* dari Random Forest berhubungan negative dengan *classification intensity* S dari sebuah *decision tree* dan berhubungan positif dengan korelasi *P* diantara decision trees, dengan begitu semakin besar *classification intensity* S, maka korelasi *P* akan semakin kecil. Semakin kecilnya batasan dari *generalization error* akan berdampak pada hasil akurasi yang lebih tinggi pada Random Forest. Rumus dari algoritma Random Forest diatas diangkat langsung dari penelitian sebelumnya pada journal (Anandarajan dkk., 2019).

2.5 Evaluasi Performa

Evaluasi akurasi dari sistem NER yang telah dibuat dievaluasi dengan cara perhitungan *Precision* (P), *Recall* (R) dan juga *f-meassure,* akurasi (Annachira Kushalappa dkk., 2015) . akurasi dari sistem ini dapat diukur dari jumlah kata yang benar di prediksi sebagai entitas dari jumlah kata yang terdapat pada artikel berita

$$Akurasi = \frac{TP + TN}{TP + TN + FP + FN} \quad \ldots(2.15)$$

TP merupakan hasil benar positif dari kata yang benar di prediksi sebagai sebuah entitas oleh sistem NER sedangkan TN adalah kata yang telah benar di prediksi bukan sebagai entitas. FP merupakan kata yang telah salah diprediksi oleh sistem NER sebagai sebuah entitas dan FN merupakan kata entitas namun salah di prediksi oleh sistem NER.

Precision merupakan pengukuran jumlah kata yang telah benar positif (TP) di prediksi oleh sistem sebagai sebuah entitas dari kata-kata yang telah ditandai dibagi dengan benar positif (TP) ditambah salah positif (FP).

$$\text{Precision} = \frac{TP}{(TP + FP)} \quad \ldots(2.16)$$

Hasil dari *Recall* didapatkan dari pengukuran jumlah kata yang telah benar positif (TP) di prediksi oleh sistem sebagai sebuah entitas dari kata-kata yang telah ditandai dibagi dengan hasil dari benar positif (TP) ditambah dengan salah negative (FN)

$$\text{Recall} = \frac{TP}{(TP + FN)} \quad \ldots(2.17)$$

Hasil dari precision dan recall kemudian digabungkan untuk membentuk F-*meassure* dari peforma sistem NER yang telah dibuat. Hasil ini dihitung dengan cara.

$$F_1 = 2 \,. \frac{precision \,.\, recall}{precision + recall} \quad \ldots(2.18)$$