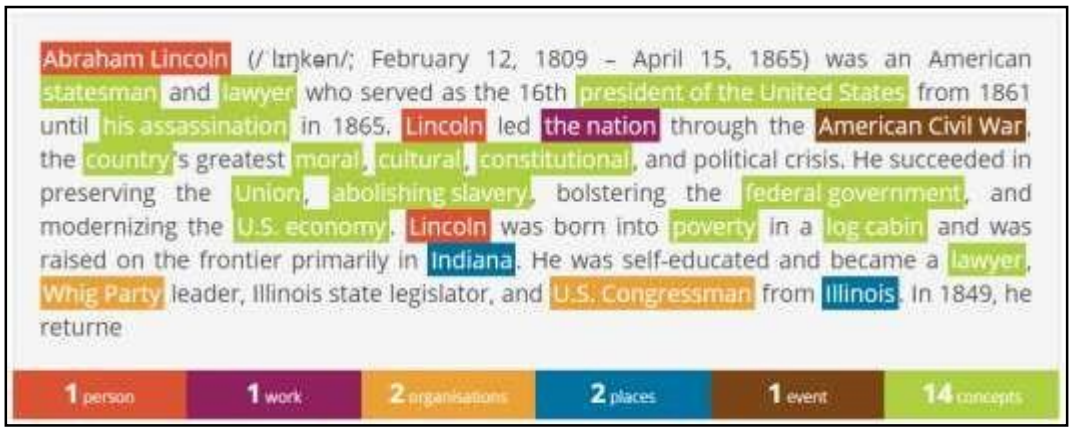
BAB 2

LANDASAN TEORI

2.1 Named Entity Recognition (NER)

Named Entity Recognition merupakan salah satu jenis teknik dari *Natural Language Processing* (NLP) yang berguna untuk mengekstraksi entitas dari sebuah teks dan mengklasifikasikannya ke dalam kategori yang telah ditentukan sebelumnya (Ines Roldos, 2020). NER banyak diterapkan pada sistem *question answering*, *information retrieval*, *co-reference resolution*, *topic modeling*, dan lain-lain (Yadav dan Bethard, 2019).



Gambar 2.1 Contoh Penerapan NER

(Sumber: dandelion.eu)

Secara umum, terdapat dua grup metode yang biasa diterapkan pada NER, yaitu *rule based* dan *machine learning*. Sistem *rule-based* berjalan pada basis *rules* yang telah ditentukan oleh pakar, sedangkan sistem *machine learning* menggunakan sejumlah data untuk mempelajari pola dari setiap data, sehingga dapat memprediksi label dari data selanjutnya (Konkol, 2015).

2.2 Preprocessing

Preprocessing merupakan tahapan awal yang diterapkan pada data *input*, dalam hal ini dalam bentuk artikel digital. *Preprocessing* terdiri dari beberapa tahapan, antara lain (Nithya, 2016; Frendy Wong, 2020):

1. Case Folding

Case Folding bertujuan untuk membuat informasi yang diterima hanya berisi huruf a-z, sedangkan karakter lainnya diubah atau dihilangkan. *Case Folding* mencakup pengubahan keseluruhan teks menjadi *lowercase*, menghapus angka, dan menghapus tanda baca pada teks *input*-an.

2. Stopword Removal

Tahapan ini bertujuan untuk menghilangkan kata penyambung atau kata pengisi dari setiap kalimat. Kata yang akan dihilangkan misalnya “dan”, “atau”, ”misalnya”, ”adalah”, ”ialah”, ”yaitu”, “tetapi”, ”di”, dan sebagainya, sehingga dihasilkan hanya kata yang memiliki makna sehingga lebih mudah dipahami.

Tabel 2.1 Contoh penerapan Stop Word Removal

Kalimat Awal	[Menurut] [Dimas] [pelajaran] [IPA] [lebih] [menarik] [dan] [menantang]
Kalimat Sesudah Diterapkan Stop Word Removal	[Dimas] [pelajaran] [IPA] [menarik] [menantang]

3. Stemming

Stemming merupakan proses untuk menghilangkan infleksi kata; mengembalikan kata menjadi bentuk dasarnya. Misalnya kata “ditulis”, “tertulis”, “menulis”, “tulisan” akan ditransformasi menjadi kata “tulis”.

Tabel 2.2 Contoh penerapan Stemming

Kalimat Awal	[Seluruh] [siswa] [diwajibkan] [untuk] [menggunakan] [seragam] [lengkap]
Kalimat Sesudah Diterapkan Stemming	[Seluruh] [siswa] [wajib] [untuk] [guna] [seragam] [lengkap]

4. Tokenization

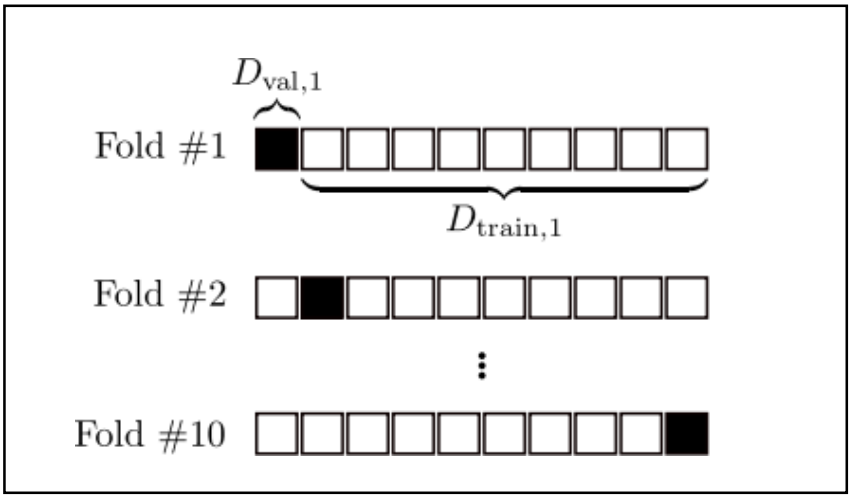
Tokenization merupakan proses pemisahan teks menjadi potongan-potongan yang disebut dengan token. Token yang dihasilkan dapat berbentuk kata maupun kalimat, namun pada penelitian ini token yang dihasilkan dalam bentuk kata.

Tabel 2.3 Contoh penerapan Tokenization

Kalimat Awal	Saya memiliki hamster bernama George dan Giadarno.
Kalimat Sesudah Diterapkan Tokenization	[Saya] [memiliki] [hamster] [bernama] [George] [dan] [Giadarno] [.]

2.3 K-Fold Cross Validation

K-Fold *Cross Validation* merupakan metode *resampling* data, di mana dataset dibagi menjadi *subsets* sejumlah k. Partisi dilakukan tanpa terjadi pengubahan. Model akan dilatih menggunakan k-1 *subset*s, sedangkan 1 *subset* yang tersisa dijadikan data *testing*. Hal tersebut dilakukan sebanyak k iterasi (*folds*), dengan pemilihan *subset testing* yang berbeda-beda. Posisi *subset* yang dijadikan data *testing* sesuai dengan nilai *fold* yang sedang berjalan. Misalnya pada *fold* pertama, maka *subset* pertama akan menjadi data *testing*, sisanya menjadi data *training*. Pada *fold* kedua, *subset* kedua menjadi data *testing* dan sisanya menjadi data *training*. Begitu seterusnya hingga keseluruhan *folds*. Hal ini menghasilkan tidak ada data *testing* yang berulang untuk setiap *fold* (Berrar, 2018).

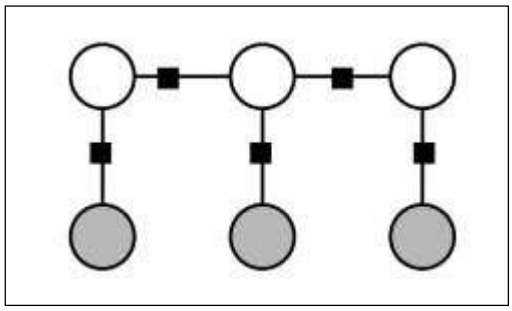


Gambar 2.2 Ilustrasi penerapan K-Fold dengan 10 folds

2.4 Conditional Random Fields (CRF)

Conditional Random Field merupakan *discriminative models* berbentuk model probabilistik dan banyak digunakan pada pelabelan atau segmentasi *sequence data*. CRF memiliki banyak kelebihan dibandingkan dengan model

probabilitas lain, misalnya *Hidden Markov Model* (HMM) dan *Maximum Entropy Markov Model* (MEMM), karena pada CRF, dapat ditentukan sendiri seberapa banyak fitur yang diinginkan, dengan bobot yang bebas (Jaariyah dan Rainarli, 2017). CRF memiliki beberapa bentuk, salah satunya yaitu *linear-chain* CRF.



Gambar 2.3 Model Linear-chain CRF

Linear-chain CRF salah satunya digunakan untuk NER, karena hasil dari NER merupakan label *sequence*. *Linear-chain* CRF memenuhi definisi sebagai berikut (Sutton dan Mccallum, 2019):

1. Diberikan X,Y vector acak, λk adalah parameter vector, dan fk(y, y0 , xt) merupakan nilai nyata *feature functions* dari k = 1 hingga k = K, di mana K adalah angka, maka *linear-chain* CRF adalah distribusi p(y|x) dalam bentuk

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

di mana Z(x) adalah fungsi normalisasi *instance-spesific*

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

Permasalahan utama *linear-chain* CRF yaitu untuk mencari parameter vektor $\lambda\rightarrow = \{\lambda 1, \ldots, \lambda k\}$ menggunakan metode *maximum likelihood* pada saat training (Frendy Wong, 2020).

2. Dimisalkan $\theta = \{\lambda_k\}$, diberikan *training* data D = $\{x(i), y(i)\}_{i=1}^{N}$ di mana setiap $x^{(i)} = \{ x_1^{(i)}, x_2^{(i)}, \ldots, x_T^{(i)}\}$ merupakan sekuen dari *input* dan setiap $y^{(i)}$

= { $y_1^{(i)}, y_2^{(i)}, \dots, y_T^{(i)}$} merupakan sekuen dari *output* yang diharapkan, serta setiap sekuen diasumsikan independen. *Parameter estimation* biasanya ditampilkan dalam bentuk *maximum likelihood*, namun karena model ini merupakan *conditional distribution*,maka akan ditampilkan dalam *log likelihood* yaitu

$$L(\theta) = \sum_{i=1}^{N} \log\ p(y^{(i)}|x^{(i)}) \quad (2.3)$$

3. Untuk memperoleh *conditional likelihood* p(y|x; θ) yaitu dengan permisalan kombinasi p(x; θ') yang membentuk p(y,x). Dan dihasilkan *joint log likelihood*

$$\log p(y, x) - \log p(y|x; \theta) + \log p(x; \theta') \quad (2.4)$$

4. Substitusi CRF model pada poin (1) ke *likelihood* pada poin (3), sehingga dihasilkan

$$L(\theta) = \sum_{i=1}^{N} \sum_{t=1}^{T} \sum_{k=1}^{K} \lambda_k\ f_k\ (y_t^{(i)}, y_{t-1}^{(i)}, x_t^{(i)}) - \sum_{i=1}^{N} \log Z(x^{(i)}) \quad (2.5)$$

5. Untuk menghindari terjadinya *overfitting*, digunakan *regularization*, yaitu sebuah penalty pada *weight vectors* yang perhitungan *norm*-nya terlalu besar. Penalti yang diterapkan berdasarkan pada *Euclidean norm* dari θ dan *regularization parameter* $1/2\sigma^2$ sebagai *strength* dari penalty tersebut. Maka dihasilkan *regularized log likelihood* sebagai

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

NER yang menggunakan CRF terbentuk dalam *undirected graphical model* dari *condirionally trained probabilistic finite state automata*. CRF digunakan untuk menghitung probabilitas kondisional dari setiap nilai yang

sudah didesain pada *output nodes* berdasarkan nilai yang diberikan pada *input nodes* (Patil, Patil dan Pawar, 2020).

2.5 BIO Format

BIO Format merupakan format untuk setiap kata. Terdapat tiga jenis potongan tag untuk melabeli setiap token yang ada, antara lain B (Beginning), I (Inside), dan O (Outside). Tag B berguna untuk menandai kata pertama dari suatu entitas. Tag I berguna untuk menandai kata kedua dan seterusnya dari sebuah entitas. Tag O berguna untuk menandai kata yang bukan merupakan suatu entitas (Tang et al., 2013; Azalia, Bijaksana dan Huda, 2019).

Selanjutnya, tag BIO tersebut diikuti dengan jenis entitasnya. Dalam penelitian ini digunakan 5 jenis entitas, sehingga dihasilkan sebelas tag seperti pada Tabel 2.4.

Tabel 2.4 Tag dengan BIO Format

Jenis Entitas	Tag
Tarian	B-Tarian dan I-Tarian
Musik	B-Musik dan I-Musik
Tokoh	B-Tokoh dan I-Tokoh
Pertunjukan	B-Pertunjukan dan I-Pertunjukan
Alat musik	B-Alat dan I-Alat
Bukan entitas lima kategori di atas	O

2.6 Limited-Memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS)

L-BFGS merupakan metode Quasi-Newton (QN) yang digunakan untuk optimasi parameter. Metode Newton sendiri berisi matriks Jacobian yang memerlukan *large-scale time* dalam pengeksekusiannya karena bersifat *derivative computation*. Oleh karena itu muncul metode Quasi-Newton yang mengganti *deritative compution* menjadi fungsi *direct computation* (Susmitha dan Haritha, 2020).

L-BFGS sendiri merupakan pengembangan dari BFGS, di mana L-BFGS memerlukan memori yang lebih kecil dibandingkan dengan BFGS. Saat BFGS menyimpan sebuah aproksimasi dense n*n ke dalam *inverse* Hessian, L-BFGS menyimpannya sebagai sebuah vector yang merepresentasikan dugaan secara implisit. Hal ini yang menyebabkan penggunaan memori pada L-BFGS jauh lebih kecil (Saputro dan Widyaningsih, 2017).

2.7 L1 dan L2 Regularization

Regularization merupakan teknik yang paling sering digunakan untuk memberikan penalti terhadap model di *machine learning*, sehingga dapat meminimalisir *overfitting* dan membuat performa model menjadi lebih baik untuk *input*-an yang baru. Model regresi yang menggunakan L1 regularization disebut dengan Lasso Regression, sedangkan model regresi yang menggunakan L2 regularization disebut Ridge Regression (Neelam Tyagi, 2021).

L1 regularization memberikan penalti sebesar nilai mutlak besarnya koefisien. Di sisi lain, L2 regularization memberikan penalti dengan nilai kuadrat dari nilai mutlak koefisien. Pemberian bobot yang tinggi untuk L1 maupun L2 dapat menyebabkan model menjadi *underfit* (Neelam Tyagi, 2021).