2. LANDASAN TEORI

2.1. Tinjauan Pustaka

Pada bab ini akan dijelaskan mengenai teori-teori yang akan digunakan dalam penulisan skripsi dan pembuatan aplikasi.

2.1.1. Natural Language Processing (NLP)

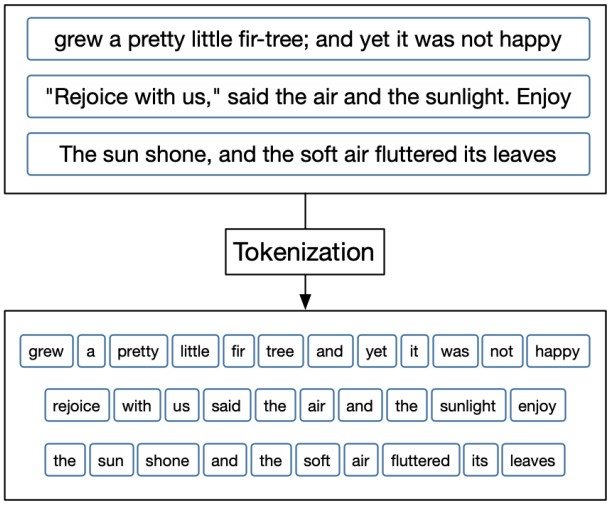
Natural Language Processing (NLP) adalah sebuah bidang dalam ilmu komputer dan ilmu bahasa yang mempelajari interaksi antara komputer dan bahasa natural manusia (Kumar, 2013). NLP mencakup studi dalam berbagai tingkatan dalam bahasa manusia, yaitu fonologi, morfologi, leksikon, sintaktik, semantik, pembicaraan, dan pragmatik (Ribeiro, 2021). Penerapan-penerapan NLP dapat berupa Word Processing, pengecekan dan perbaikan ejaan, Information Retrieval (IR), Information Extraction, Information Categorization, penjawaban pertanyaan, Information Summarization, dan Machine Translation (Chowdhary, 2020). Dalam penelitian ini, teknik-teknik NLP dipakai untuk memroses data transkrip yang didapatkan dari sebuah diskusi melalui program *transcription*.

2.1.2. Text Preprocessing

Dalam NLP, *Text Preprocessing* adalah teknik yang digunakan untuk mempersiapkan teks yang tidak terstruktur menjadi data yang baik dan siap untuk diolah (Katryn, 2021). *Preprocessing* penting untuk dilakukan untuk mengurangi ukuran file teks dokumen yang akan dianalisis (Gurusamy et al., 2014). Penggunaan teknik-teknik preprocessing dilakukan sesuai dengan karakteristik data teks dan kebutuhan penelitian. Teknik-teknik yang digunakan dalam penelitian ini antara lain *tokenization*, *stemming, lemmatization*, dan *stop word removal*.

Tokenization adalah sebuah proses memecah sebuah dokumen menjadi kumpulan kata, frasa, simbol, atau elemen-elemen lainnya yang disebut token (Gurusamy et al., 2014). Secara umum, ada tiga tipe token yang bisa didapatkan dari sebuah dokumen, yaitu *word*, *character*, dan *subword* (Pai, 2020). Token bertipe *word* memecah sebuah dokumen menjadi sekumpulan kata, dan *subword* memecah sebuah dokumen menjadi sekumpulan kata yang dipecah ke bentuk dasar dan imbuhannya. Untuk penelitian ini, dilakukan *word-tokenization*, yaitu pemecahan dokumen menjadi kumpulan kata-kata yang akan direpresentasikan dalam vektor

pada proses selanjutnya. Proses *word-tokenization* ini juga dilakukan untuk mempermudah *lemmatizing* dan *stemming* yang akan dilakukan sebagai bagian dari *preprocessing*.



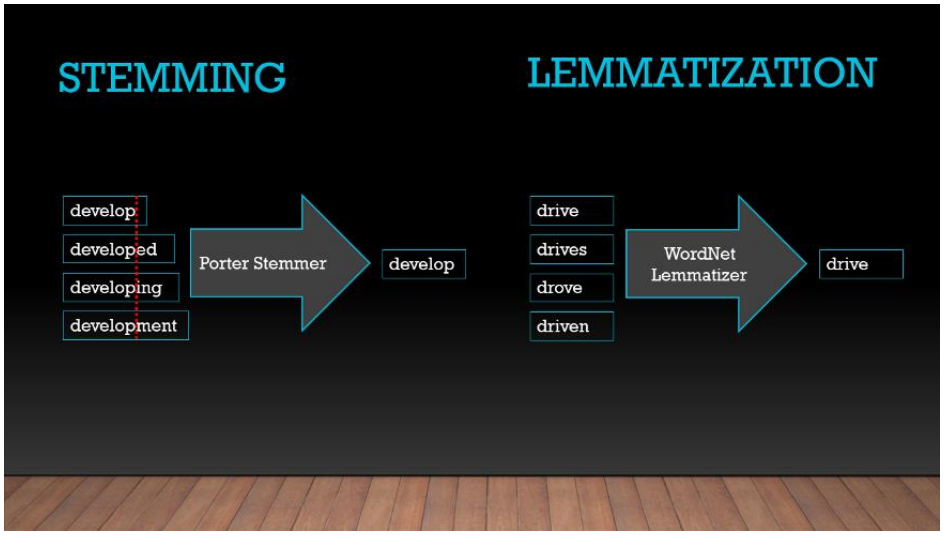
Gambar 2.1 Ilustrasi *word tokenization*

Sumber: Hvitfeld, E., & Silge, J. (2022, May 11). *Chapter 2 Tokenization | Supervised Machine Learning for Text Analysis in R*. Supervised Machine Learning for Text Analysis in R. https://smltar.com/tokenization.html

Stemming adalah sebuah proses mengubah varian sebuah kata menjadi kata dasarnya, yang disebut *stem* atau batang (Gurusamy et al., 2014). Teknik ini dilakukan untuk generalisasi kata kerja, supaya kata-kata kerja dalam bentuk bentuk berbeda tidak dianggap token-token yang berbeda. Dalam penelitian yang diajukan, karena keseluruhan data transkrip ada dalam Bahasa Inggris, dapat diterapkan algoritma Porter Stemming yang diterapkan oleh Martin Porter. Algoritma ini memanfaatkan beberapa prinsip dan aturan untuk menghilangkan *suffix* (akhiran) pada setiap kata untuk mendapatkan stem dari kata tersebut. Algoritma ini dimaksudkan untuk meningkatkan kinerja *information retrieval* dan mungkin memproduksi sebuah *stem* yang sama untuk dua kata dengan makna yang berbeda (Porter, 1980).

Lemmatization adalah proses pengelompokan bentuk-bentuk terinfleksi dari sebuah kata menjadi satu bentuk dasar (*lemma*) ("Lemmatize Definition and Meaning | Collins English Dictionary," 2023). Tidak seperti *stemming* yang menggunakan pendekatan berdasarkan karakter, pendekatan reduksi dengan *lemmatization* bergantung pada pengetahuan tata bahasa yang bersangkutan untuk mengenali *lemma* sebuah kata infleksi. Oleh karena itu, proses ini membutuhkan proses *POS tagging* untuk mengenali bentuk kata (contohnya: "working" sebagai kata kerja atau sebagai kata sifat –tergantung konteks kalimat) untuk melakukan klasifikasi

dengan benar. Dengan pendekatan ini, makna kata dapat dipertahankan dan tidak ambigu dengan *lemma* lainnya. Dalam penelitian yang akan dijalankan, akan diuji pendekatan *stemming* dan *lemmatization* secara terpisah untuk mendapatkan hasil yang lebih baik.



Gambar 2.2 Perbandingan pendekatan *stemming* dan *lemmatization*

Sumber: *Stemming vs Lemmatization NLP | Kaggle*. (n.d.). https://www.kaggle.com/getting-started/186152

Stop word removal adalah sebuah proses penghapusan kata-kata yang frekuensi kemunculannya tinggi, namun tidak memberikan makna yang signifikan pada dokumen. *Stop word* dalam Bahasa Inggris dapat mencapai 20 hingga 30 persen jumlah kata total dalam sebuah teks dokumen (Gurusamy et al., 2014). Penghapusan *stop-words* pada sebuah korpus bertujuan untuk mengurangi *noise* dan token-token kata yang redundan.

Sample text with Stop Words	Without Stop Words
GeeksforGeeks – A Computer Science Portal for Geeks	GeeksforGeeks , Computer Science, Portal ,Geeks
Can listening be exhausting?	Listening, Exhausting
I like reading, so I read	Like, Reading, read

Gambar 2.3 *Stop word removal* pada sejumlah kalimat Bahasa Inggris

Sumber: GeeksforGeeks. (2023). Removing stop words with NLTK in Python. *GeeksforGeeks*. https://www.geeksforgeeks.org/removing-stop-words-nltk-python/

2.1.3. Density-Based Clustering

Clustering atau *cluster analysis* adalah sebuah metode *unsupervised machine learning* yang bertujuan untuk menemukan hubungan dan pengelompokan yang sudah ada pada data (*10 Clustering Algorithms With Python, 2020b*). Metode ini disebut *unsupervised* karena tidak membutuhkan label pada data; *clustering* berfungsi untuk mengelompokkan data dan menarik kesimpulan berdasarkan kemiripan-kemiripan fitur data tersebut. Ada beberapa pendekatan yang bisa diambil untuk melakukan *clustering*, diantaranya *centroid-based clustering*, *density-based clustering*, *distribution-based clustering*, dan *hierarchical clustering* ("Clustering Algorithms," n.d.). Masing-masing pendekatan tersebut. Konsep fundamental *clustering* adalah mengidentifikasikan kelompok-kelompok yang terbentuk dari kemiripan antar rekor data. Terutama untuk data kuantitatif, digunakan fungsi yang menentukan jarak antar poin data yang dimiliki (Xu & Tian, 2015). Terdapat beberapa algoritma untuk menentukan jarak yang dipakai dalam *machine learning*, diantaranya adalah Euclidian Distance, Cosine Distance, dan Mahalanobis Distance.

Minkowski distance $$\left(\sum_{l=1}^{d} |x_{il} - x_{jl}|^n\right)^{1/n}$$

Cosine distance $$1 - \cos\alpha = \frac{x_i^T x_j}{\|x_i\|\|x_j\|}$$

Gambar 2.4 Beberapa fungsi jarak

Sumber: Xu, D., & Tian, Y. (2015). A Comprehensive Survey of Clustering Algorithms. *Annals of Data Science, 2*(2), 165-193. https://link.springer.com/article/10.1007/s40745-015-0040-1

Pendekatan yang diambil untuk suatu kasus tertentu harus sesuai dengan kebutuhan penelitian dan sifat dari dataset yang dimiliki, dan oleh *tools* yang tersedia untuk melakukan *clustering* (Witten et al., 2016). Dalam penelitian yang diajukan, akan dilakukan *anomaly detection* dengan mengidentifikasikan outlier pada data. Oleh karena itu, *density-based clustering* akan digunakan karena tidak memasukkan *outlier* ke *cluster* manapun. ("Clustering Algorithms," n.d.), dan menentukan cluster-cluster berdasarkan kedekatan relatif antar poin data. Untuk tujuan ini, akan digunakan metode DBSCAN (Density Based Spatial Clustering of Applications with Noise) untuk memproduksi *cluster-cluster* pada dataset, dengan harapan

ditemukannya poin-poin data yang tidak tergabung dalam cluster manapun sebagai *outlier.* Adapun akan diuji secara paralel fungsi-fungsi jarak yang ada untuk menentukan pendekatan yang paling efektif dan sesuai dengan kebutuhan penelitian.

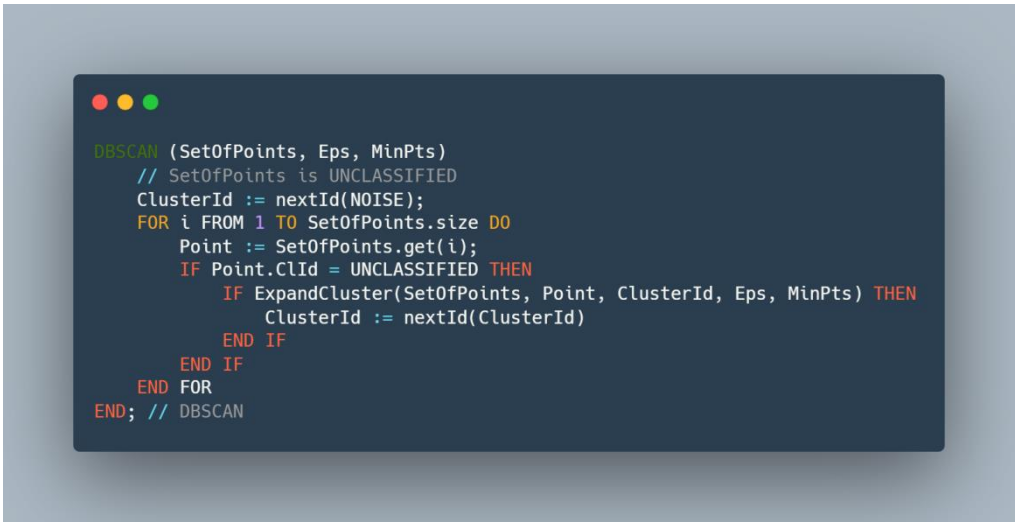
2.1.4. *Density-Based Spatial Clustering of Applications with Noise* (DBSCAN)

DBSCAN adalah sebuah algoritma *density-based clustering* yang membuat *cluster-cluster* pada poin-poin data yang tidak berlabel berdasarkan kedekatannya. Algoritma ini dibuat sebagai realisasi pendekatan intuitif yang dapat disimpulkan ketika mengevaluasi poin-poin data yang memiliki kedekatan (*density*) tertentu yang dapat memisahkan sebuah *cluster* dengan *noise* ("A Density-based Algorithm for Discovering Clusters in Large Spatial Databases With Noise," 1996).

Algoritma DBSCAN menerima dua parameter yang ditentukan oleh pengguna, yaitu *MinPts* dan ε (eps; epsilon).

Adapun poin-poin data dikategorikan berdasarkan parameter yang ada sebagai berikut:

- *Core point*, jika ada *MinPts* buah poin data yang terletak dalam radius ε di sekelilingnya.
- Poin data *p* dikatakan *directly density-reachable point* dari poin data *q*, jika poin data *p* memiliki jarak maksimum ε dari *q.*
- Poin data *p* dikatakan *density-reachable point* dari poin data *q*, jika ada sebuah jalur atau path di antara poin data *p* dan *q* berupa poin-poin data yang berjarak maksimum ε antar satu sama lain.
- *Outlier* atau *noise point*, jika poin data tersebut tidak reachable dari poin-poin data lainnya.



```
DBSCAN (SetOfPoints, Eps, MinPts)
    // SetOfPoints is UNCLASSIFIED
    ClusterId := nextId(NOISE);
    FOR i FROM 1 TO SetOfPoints.size DO
        Point := SetOfPoints.get(i);
        IF Point.ClId = UNCLASSIFIED THEN
            IF ExpandCluster(SetOfPoints, Point, ClusterId, Eps, MinPts) THEN
                ClusterId := nextId(ClusterId)
            END IF
        END IF
    END FOR
END; // DBSCAN
```

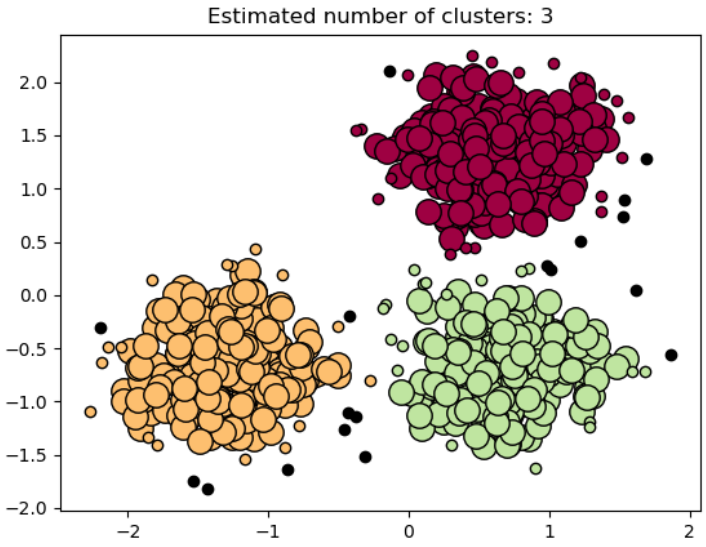
Gambar 2.5 Pseudocode Penerapan DBSCAN

```
ExpandCluster (SetOfPoints, Point, ClId, Eps,MinPts) : Boolean;
    seeds:=SetOfPoints.regionQuery(Point,Eps);
    IF seeds.size<MinPts THEN // no core point
        SetOfPoint.changeClId(Point,NOISE);
        RETURN False;
    ELSE    // all points in seeds are density-
            // reachable from Point
        SetOfPoints.changeClIds(seeds,ClId);
        seeds.delete(Point);
        WHILE seeds <> Empty DO
            currentP := seeds.first();
            result := SetOfPoints.regionQuery(currentP,Eps);
            IF result.size >= MinPts THEN
                FOR i FROM 1 TO result.size DO
                    resultP := result.get(i);
                        IF resultP.ClId IN {UNCLASSIFIED, NOISE} THEN
                            IF resultP.ClId = UNCLASSIFIED THEN
                                seeds.append(resultP);
                            END IF;
                            SetOfPoints.changeClId(resultP,ClId);
                        END IF; // UNCLASSIFIED or NOISE
                END FOR;
            END IF; // result.size >= MinPts
            seeds.delete(currentP);
        END WHILE; // seeds <> Empty
        RETURN True;
    END IF
END; // ExpandCluster
```

Gambar 2.6 Pseudocode Fungsi ExpandCluster() yang Dinyatakan dalam Pseudocode DBSCAN

DBSCAN diterapkan secara iteratif untuk setiap poin yang ada dalam dataset. Setiap iterasi menentukan apakah diperlukan adanya *cluster* baru atau tidak, dengan cara menghitung poin-poin yang berdekatan dengan poin data yang bersangkutan. Dalam Gambar 2.5., perhitungan dilakukan dengan menjalankan fungsi ExpandCluster() yang dijelaskan dalam Gambar 2.6. Dalam fungsi ExpandCluster(), akan dicari poin-poin yang merupakan *core point*s. Poin-poin tersebut akan diiterasi secara rekursif untuk mencari dan menandai *density-reachable points* yang ada di sekitarnya. Poin-poin yang tidak memenuhi kriteria akan ditandai sebagai *noise* atau *outlier.*

DBSCAN mendefinisikan sebuah *cluster* sebagai kumpulan poin-poin data yang *reachable* antar satu sama lain. *Outlier* adalah poin data yang tidak *reachable* dari poin-poin data lainnya. Oleh karena itu, DBSCAN akan digunakan dalam penelitian ini untuk mengidentifikasikan anomali dalam data berupa data point yang dianggap *noise* atau *outlier*.



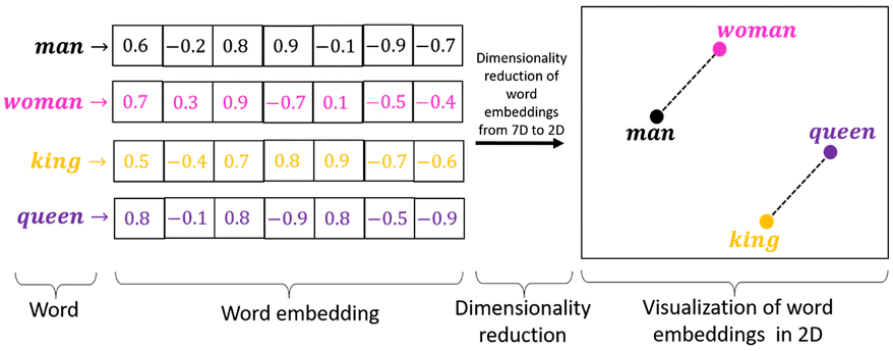
Gambar 2.7 Clustering dengan DBSCAN; poin-poin data hitam menunjukkan *outlier*

Sumber: *Comparing different clustering algorithms on toy datasets*. (n.d.). Scikit-learn. https://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_comparison.html

2.1.5. Word Embedding

Word embedding adalah metode untuk menangkap makna semantik dan sintaktik kata-kata yang ada dalam sebuah korpus teks yang tidak dilabeli (Lai et al., 2016). Metode ini dilakukan untuk melakukan *encoding* setiap word token yang telah dikumpulkan menjadi sebuah vektor bilangan riil yang bisa direpresentasikan dalam sebuah dimensi tertentu. Vektor-

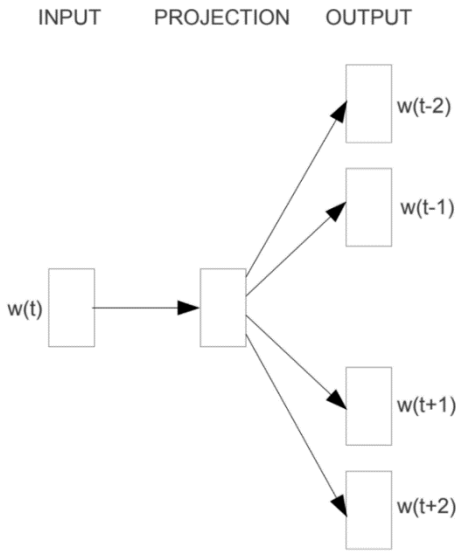
vektor bilangan yang didapatkan dari hasil *embedding* dapat ditransformasikan menjadi vektor dengan dimensi lebih kecil melalui sebuah proses yang disebut *dimensionality reduction*.



Gambar 2.8 Visualisasi *Word2Vec.*

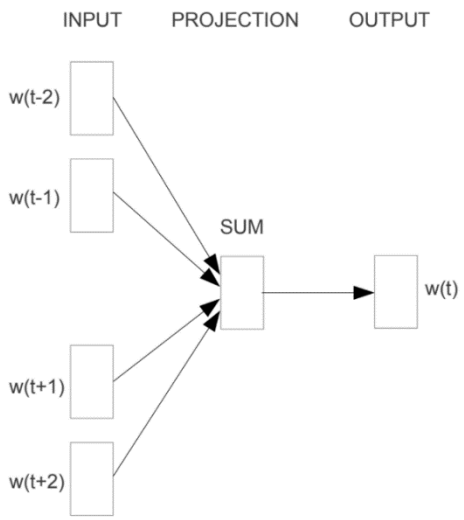
Sumber: Gautam, H. (2020, March 1). Word Embedding: Basics - Hariom Gautam - Medium. *Medium*. Retrieved December 16, 2022, from https://medium.com/@hari4om/word-embedding-d816f643140

Word2Vec adalah suatu metode untuk membangun *Word Embedding* secara efisien berupa *neural network* yang terdiri dari dua *layer* untuk melakukan proses pengubahan teks menjadi vektor. *Word2Vec* memiliki 2 pilihan arsitektur yaitu *CBOW* (*Continuous Bag of words*) dan *Skip-Gram*.



Gambar 2.9 Arsitektur *skip-gram Word2Vec*

Sumber: Firdaus, A. (2019, August 13). Cara Kerja Word2Vec. *Medium*. Retrieved December 20, 2022, from https://medium.com/@afrizalfir/mengenal-word2vec-af4758da6b5d



Gambar 2.10 Arsitektur CBOW *Word2Vec***.**

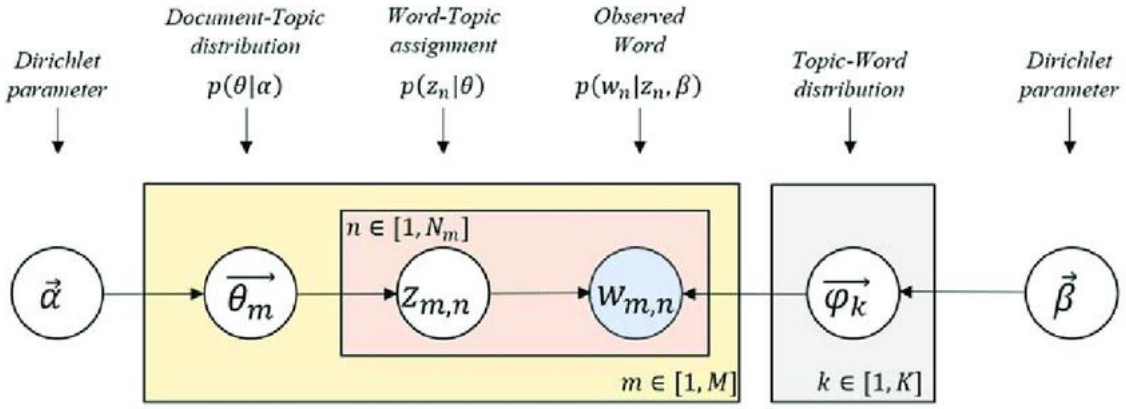
Sumber: Firdaus, A. (2019, August 13). Cara Kerja Word2Vec. *Medium*. Retrieved December 20, 2022, from https://medium.com/@afrizalfir/mengenal-word2vec-af4758da6b5d

Arsitektur *skip-gram* memprediksi kata-kata konteks yang ada di sekitar kata utama. Sebaliknya, arsitektur *CBOW* mencoba memprediksi *encoding* sebuah kata berdasarkan kata-kata konteks di sekitarnya. Dengan arsitektur-arsitektur ini, *Word2Vec* memprediksi konteks sebuah word token dalam korpus berdasarkan sejumlah kata-kata konteks yang berdekatan dengan kata pilihan; banyaknya kata yang diambil sebagai konteks disebut dengan istilah *window*. Dalam proses *training*, pendekatan ini akan berusaha mengatur *weight* dari setiap kata sedekat mungkin dengan makna aslinya. Kata-kata yang dekat maknanya akan memiliki vektor representasi yang berdekatan dalam *embedding Word2Vec* (Firdaus, 2019). Dalam studi yang akan dilakukan, sebuah dokumen terdiri dari kumpulan *word-token*. Akan dilakukan *sentence-embedding* untuk mengagregasikan word-embedding untuk setiap token dan mendapatkan sebuah vektor n-dimensi yang merepresentasikan sebuah dokumen.

2.1.6. *Topic Modelling*

Topic Modelling adalah sebuah algoritma yang mengidentifikasikan pola-pola tersemat dalam kata-kata dalam sekumpulan dokumen (Jacobi et al., 2015). Proses ini merupakan sebuah penerapan *unsupervised machine learning* yang digunakan untuk memodelkan topik-topik yang terkandung dalam sebuah dokumen secara probabilistik. Salah satu *topic modelling* yang populer adalah *Latent Dirichlet Allocation* (LDA). Dalam penelitian yang diajukan ini, distribusi

topik per dokumen yang didapatkan dari proses *topic modelling* dengan LDA akan dianggap sebagai fitur-fitur data.



Gambar 2.11 Diagram Model LDA

Sumber: Lee, J., Kang, J. Y., Jun, S., Lim, H., Jang, D., & Park, S. (2018). Ensemble Modeling for Sustainable Technology Transfer. *Sustainability*, *10*(7), 2278. https://doi.org/10.3390/su10072278

Keterangan:

- α, β : distribusi Dirichlet
- m : dokumen ke-m dalam M buah dokumen
- M : jumlah total dokumen
- n : kata ke-n dalam N buah kata
- N : jumlah total kata
- k : topik ke-k dalam K buah topik
- K : jumlah total topik
- θ_m : distribusi topik dalam dokumen ke-m
- ϕ_k : distribusi kata dalam topik ke-K
- $z_{m,n}$: topik yang didapatkan dari θ_m yang diberikan ke kata ke-n
- $w_{m,n}$: kata ke-n dalam dokumen ke-m

LDA adalah sebuah model statistika generatif yang dibuat untuk mengekstraksi variabel-variabel yang tersemat untuk mencari kemiripan dalam data. Metode ini dilakukan dengan mengambil secara stokastik sampel-sampel topik dari kumpulan topik per dokumen serta kata dari kumpulan kata per dokumen, lalu menggabungkannya dalam sebuah dokumen

baru yang mengandung kata-kata dan topik yang mencerminkan dokumen awalnya (Blei et al., 2001). Dalam satu iterasi untuk generasi topik LDA, akan dicari distribusi kata-kata untuk tiap topik pada kumpulan topik K, dengan memperhatikan distribusi Dirichlet β. Akan juga dicari distribusi topik-topik untuk tiap dokumen pada kumpulan dokumen M dengan memperhatikan distribusi Dirichlet α. Untuk setiap dokumen m pada kumpulan dokumen M, akan diberikan sebuah topik yang dihasilkan dari distribusi dokumen-topik sebelumnya, dan sebuah kata yang didapatkan dari distribusi topik-kata sebelumnya, untuk menginferensikan distribusi topik per dokumen.

Proses ini akan diiterasi sebanyak yang diinginkan, untuk menghasilkan konvergensi statistika. Dalam penelitian yang akan dilakukan, akan diuji variasi jumlah topik yang akan menghasilkan distribusi topik dengan dimensi yang berbeda-beda. Diharapkan ditemukan jumlah topik yang optimal, sehingga menciptakan vektor fitur yang reflektif atas setiap dokumen.

2.1.7. TF-IDF

Term Frequency—Inverse Document Frequency (TF-IDF) adalah sebuah metode untuk memberikan *weight* sebuah kata dalam sebuah dokumen dalam sebuah korpus. Metode ini menggabungkan nilai *Term Frequency* (TF) yang mengevaluasi seberapa sering sebuah kata muncul dalam sebuah dokumen dibandingkan kata-kata lainnya, dan nilai *Inverse Document Frequency* (IDF) yang mengekspresikan perbandingan terbalik (inverse) jumlah dokumen dimana *term* atau kata tertentu muncul (Salton & Buckley, 1988).

TF dinyatakan dalam rumus sebagai berikut :

$$TF_{(t,d)} = \frac{f_{(t,d)}}{\sum_{(t',d)} f_{(t',d)}} \tag{2.1}$$

Dimana:

- t adalah *term* atau kata tertentu
- d adalah dokumen tertentu
- $TF_{(t,d)}$ adalah *term frequency* kata t dalam dokumen d.
- $f_{(t,d)}$ adalah jumlah iterasi kata t dalam dokumen d
- $\sum_{(t',d)} f_{(t',d)}$ adalah jumlah seluruh kata yang ada dalam dokumen d

IDF dinyatakan dalam rumus sebagai berikut:

$$IDF_{(t,D)} = log \frac{N}{|\{d \in D : t \in d\}|} \quad (2.2)$$

Dimana :

- $IDF_{(t,D)}$ adalah *inverse document frequency* kata t dalam kumpulan dokumen D
- N adalah jumlah dokumen dalam korpus
- $|\{d \in D : t \in D\}|$ adalah jumlah cacah dokumen d dalam kumpulan dokumen D, dimana kata t ada dalam dokumen d

TF-IDF dinyatakan dalam matriks hasil *dot-product* sebagai berikut:

$$TFIDF_{(t,d)} = TF_{(t,d)} \cdot IDF_{(t,D)} \quad (2.3)$$

Dalam penelitian yang akan dilakukan, *term-document matrix* yang dihasilkan dari proses TF-IDF untuk *corpus* yang akan dikumpulkan berfungsi sebagai augmentasi untuk *feature extraction*. *Term-document matrix* akan digunakan sebagai *weight* untuk proses agregasi kumpulan *word-embedding* dalam sebuah dokumen untuk menghasilkan sebuah *sentence-embedding* yang mewakili dokumen tersebut. Dalam LDA, *term-document* matrix akan digunakan sebagai input inisial yang akan mempengaruhi distribusi dan sampling untuk *topic-document matrix* dan *word-topic matrix*.

2.1.8. Local Outlier Factor

Local Outlier Factor (LOF) adalah sebuah algoritma *unsupervised learning* yang digunakan untuk mendeteksi anomali pada sekelompok data. Algoritma ini mendeteksi anomali berdasarkan densitas atau kerapatan lokal sebuah rekor data dengan sejumlah rekor-rekor data yang ada di sekitarnya. Karena adanya asumsi kerapatan lokal tersebut, perlu dicari sebuah parameter berupa yang melambangkan asumsi jumlah rekor-rekor data yang ada di sekitar setiap rekor data yang dievaluasi. Bilangan ini umumnya lebih besar dari jumlah minimum anggota sebuah *cluster*, supaya dapat terbentuk *outlier-outlier* lokal terhadap *cluster-cluster* yang terbentuk, serta lebih kecil dari jumlah maksimum rekor-rekor data yang memiliki kemungkinan menjadi *outlier* (*Outlier Detection With Local Outlier Factor (LOF)*, n.d.)

Untuk setiap rekor data dalam dataset yang ada, akan dicari rekor-rekor data yang *reachable* dari rekor data tersebut dengan pendekatan *k-nearest neighbors*, yaitu pencarian rekor-rekor data dengan k buah *distance* yang paling dekat, menurut metode pengukuran jarak yang digunakan. Dalam pendekatan ini, mungkin terdapat lebih dari k buah *neighbor* yang terdeteksi relatif pada sebuah rekor data. Akan dicari *reachability distance* dari setiap rekor data tersebut, yang merupakan maksimum antara jarak rekor data dengan rekor data dan jarak rekor

data awal dengan rekor data ke-k. Setelah itu, akan dicari *local reachability density*, yaitu invers dari rata-rata *reachability distance* rekor-rekor data yang paling dekat dengan rekor data yang diuji. Setelah *local reachability density* diperoleh untuk setiap rekor data, akan dicari *local outlier factor* untuk setiap rekor data dengan cara memperoleh rata-rata *local reachability density* setiap rekor data yang paling dekat, dan membaginya dengan *local reachability* rekor data yang sedang dianalisis. Nilai-nilai local outlier factor yang telah didapatkan ini dapat diinterpretasikan sebagai berikut:

- Jika LOF dari sebuah rekor data adalah 1, maka rekor data tersebut memiliki densitas yang sama dengan rekor-rekor data tetangganya.
- Jika LOF dari sebuah rekor data lebih besar dari 1, maka rekor data tersebut memiliki densitas yang lebih tinggi dari rekor-rekor data tetangganya.
- Jika LOF dari sebuah rekor data lebih kecil dari 1, maka rekor data tersebut memiliki densitas yang lebih rendah dari rekor-rekor data tetangganya.

Menurut interpretasi tersebut, semakin kecil nilai LOF dari sebuah rekor data, semakin besar kemungkinan rekor data tersebut adalah sebuah *outlier*/anomali. Dalam penelitian ini, LOF digunakan sebagai algoritma pembanding untuk mendapatkan anomali-anomali dalam data. Akan diberikan *threshold* tertentu untuk menginterpretasikan hasil algoritma LOF dan mengidentifikasikan anomali di dalamnya.

Reachability distance dihitung untuk *k-nearest neighbors* dari sebuah rekor data sebagaimana dinyatakan oleh rumus berikut:

$$reachability\_distance_k(A,B) = \max\big(k\_distance(B), d(A,B)\big) \tag{2.4}$$

Dimana:

- k menyatakan *neighbor* ke-k dari rekor data yang bersangkutan.
- A adalah rekor data yang termasuk *k-nearest neighbors* dari B.
- B adalah rekor data yang sedang dievaluasi.
- $k\_distance(B)$ adalah jarak terbesar rekor data B dengan *k-nearest neighbors*-nya.
- $d(A,B)$ menyatakan jarak antara rekor data A dan B menurut fungsi jarak yang digunakan.

Local reachability density yang dicari untuk setiap rekor data dinyatakan dalam rumus berikut:

$$lrd_k(A) = \frac{|N_k(A)|}{\sum_{B \in N_k(A)} reachability\_distance_k(A,B)} \tag{2.5}$$

Dimana:

- $N_k(A)$ adalah jumlah *k-nearest neighbors* dari A.

Dan *local outlier factor score* untuk setiap rekor data dalam dataset dinyatakan sebagai berikut:

$$LOF_k(A) = \frac{\sum_{B \in N_k(A)} lrd_k(B)}{|N_k(A)| \cdot lrd_k(A)} \quad (2.6)$$

2.1.9. Isolation Forest (IF)

Isolation Forest adalah algoritma *anomaly detection* yang berusaha mendeteksi anomali dengan *binary decision tree.* Algoritma ini memiliki *time complexity* yang linear (Liu et al., 2008). Isolation Forest menjalankan secara independen sejumlah algoritma Isolation Tree, yaitu sebuah algoritma yang melakukan partisi biner secara rekursif, untuk melakukan partisi data menjadi dua bagian berdasarkan sebuah *feature* yang diambil secara acak, hingga semua fitur data terisolasi. Akan ditentukan sebuah *anomaly score* untuk setiap rekor data berdasarkan pada iterasi mana rekor data tersebut terisolasi. Semakin cepat sebuah rekor data terisolasi, maka semakin besar kemungkinan rekor data tersebut adalah sebuah *outlier*. *Anomaly score* yang didapatkan oleh setiap Isolation Tree akan diagregasikan, menghasilkan sebuah *anomaly score* yang dapat diinterpretasikan untuk menghasilkan *outlier*/anomali.

Rumus *decision function* untuk menyatakan *anomaly score* sebuah rekor data adalah sebagai berikut:

$$S(x, m) = 2^{\frac{-E(h(x))}{c(m)}} \quad (2.7)$$

Dimana:

- x adalah rekor data yang sedang dievaluasi
- m adalah dataset yang berisi sekumpulan rekor data.
- $h(x)$ adalah kedalaman yang diperlukan untuk mengakses rekor data ke-x
- $E(h(x))$ merupakan agregat (dalam bentuk rata-rata) dari $h(x)$ untuk setiap Isolation Tree.
- $c(m)$ merupakan panjang rata-rata dari pencarian-pencarian rekor data yang gagal, dinyatakan dalam $c(m) = 2H(m-1) - \frac{2(m-1)}{m}$

- H adalah bilangan harmonik Euler yang dinyatakan dalam $H_i = \ln(i) + \gamma$ dengan $\gamma = 0.5572156649$

2.2. Tinjauan Studi

Berikut merupakan penelitian yang telah dilakukan sebelumnya :

2.2.1. *An evaluation of document clustering and topic modelling in two online social networks: Twitter and Reddit* (Curiskis et al., 2020)

Penelitian sebelumnya yang dilakukan oleh (Curiskis et al., 2020) melakukan perbandingan *clustering* dan *topic modelling* pada data *tweet* dari media sosial Twitter dan *thread* dari media sosial Reddit. Dataset direpresentasikan dalam empat representasi fitur, yaitu *TF-IDF*, *word2vec* dengan *weight*, word2vec tanpa *weight*, dan *doc2vec.* Dilakukan teknik *clustering* pada data dengan empat metode, yaitu *K-means Clustering*, *K-medoids clustering*, *hierarchical agglomerative clustering*, dan *Non-negative matrix factorization*. Kesuksesan hasil *clustering* dalam penelitian ini diukur dengan NMI, AMI, dan ARI. Ditemukan bahwa performa paling baik didapatkan oleh representasi fitur dengan *word2vec* dan metode *k-means clustering*.

Penelitian yang diajukan memiliki beberapa perbedaan dengan penelitian yang dilakukan oleh (Curiskis et al., 2020). Pada penelitian yang diajukan, setiap rekor data berupa pernyataan peserta diskusi yang mungkin berisi lebih dari satu kalimat, sedangkan sebuah rekor data *tweet* pada *Twitter* tidak melebihi 255 karakter. Oleh karena itu, diatas *word-embedding word2vec*, perlu juga dilakukan *sentence embedding*. Dalam penelitian yang akan dilakukan akan dicoba dua variasi *feature extraction*, yaitu *word-* dan *sentence-embedding*, serta *topic modelling* dengan LDA. Fitur-fitur data akan menjadi input algoritma *density-based clustering* dengan asumsi pengetahuan dalam *domain* yang mencukupi, dan dengan harapan bahwa hasil *clustering* akan mencerminkan cluster-cluster yang ada dalam data dan memproduksi *outlier*.

2.2.2. *A Comparison of LSA and LDA for the Analysis of Railroad Accident* (Williams & Betak, 2019)

Penelitian selanjutnya (Williams & Betak, 2019) membandingkan performa *topic modelling* LDA dengan LSA, dan menemukan bahwa kedua metode menemukan beberapa topik yang eksklusif satu sama lain. Penelitian ini juga menyatakan bahwa algoritma LDA direkomendasikan untuk dataset yang berukuran lebih kecil. Dalam penelitian yang akan diajukan, LDA akan digunakan untuk mengekstraksi topik dari setiap dokumen yang berupa pendapat peserta diskusi. Distribusi topik dalam tiap dokumen akan dianggap sebagai fitur-fitur

data yang menjadi basis *density-based clustering* untuk mencari poin-poin data yang merupakan *outlier*.

2.2.3. *Anomaly Detection: A Survey* (Chandola et al., 2009)

Survei yang dilakukan oleh Chandola et al. (2009) menyatakan beberapa metode yang dapat dilakukan untuk melakukan *anomaly detection*, diantaranya adalah penerapan *density-based clustering* untuk mengelompokkan data dalam beberapa *cluster*. Poin-poin data yang menjadi *outlier* diidentifikasikan sebagai titik-titik anomali. Dalam penelitian yang diajukan, akan dilakukan *density-based clustering* pada vektor-vektor fitur data yang telah didapatkan dari fase *feature extraction* yang telah dilakukan. *Outlier-outlier* yang terbentuk oleh *clustering* yang dilakukan akan diidentifikasikan sebagai anomali dalam data dan diuji validitasnya.