1. **Hitung Bobot Kata dengan TF-IDF (*Term Frequency – Inverse Data Frequency)***

* Lakukan *preprocessing* terlebih dahulu untuk membuang kata yang tidak penting
* Menghitung frekuensi kemunculan kata dari suatu dokumen. Frekuensi tersebut dihitung dengan membandingkan kata dengan setiap kata yang sebelumnya telah di*preprocessing*. Berikut adalah rumus dari *term frequency* :

Contoh :

D1 : “Saya sedang belajar NLP, karena besok tugas NLP dikumpulkan.

Maka ,

tf(“NLP”, D1) = 2

(Lihat dari jumlah kata NLP yang ada di kata tersebut)

Selain contoh di atas, dapat juga dihitung dengan :

1. **TF biner (binary TF)**

Mencari tahu apakah suatu kata ata term terdapat didalam dokumen atau tidak, akan bernilai 1 kalau ada, kalau tidak maka bernilai 0 (seperti contoh di atas)

1. TF murni

TF yang diberikan sesuai dengan jumlah kemunculan term atau kata, jika muncul 2 maka akan bernilai 5 kata tersebut

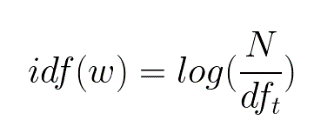
1. TF logaritmik: untuk menghindari suatu dokumen yang memiliki sedikit term namun memiliki frekuensi tinggi dalam query
2. TF normalisasi: menggunakan perbandingan antara frekuensi sebuah term dengan nilai maksimum dari keseluruhan term pada dokumen. Normalisasi ini digunakan untuk menghindari adanya bias. Gambar di bawah menunjukkan rumus dari TF normalisasi



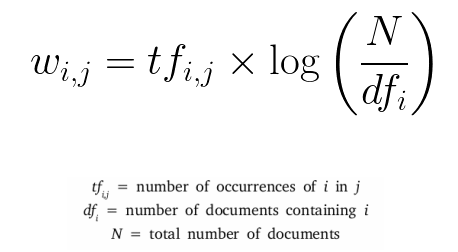
Dimana a adalah nilai antara 0 dan 1 dan umumnya 0,4, tapi biasanya juga menggunakan nilai 0,5.

* Menghitung IDF (*Inverse Data Frequency)*

IDF digunakan untuk menghitung keterkaitan sebuah *term* dengan keseluruhan dokumen. Semakin sedikit jumlah dokumen yang mengandung term yang dimaksud, maka nilai IDF semakin besar. Berikut merupakan rumus dari IDF :



* Menghitung berat/nilai dari sebuah kata dalam sebuah dokumen

Singkatnya, untuk mendapatkan berat, lakukan perkalian antara *term frequency* (TF) dan *inverse data frequency* (IDF)

Berikut merupakan contoh perhitungan untuk lebih jelasnya :

Kalimat 1: Adik pergi ke sekolah naik sepeda.

Kalimat 2: Kakak pergi ke sekolah naik motor.

|  |  |  |  |
| --- | --- | --- | --- |
| **Kata** | **TF** | **IDF** | **TF\*IDF** |
| **Kalimat 1** | **Kalimat 2** |  | **Kalimat 1** | **Kalimat 2** |
| Adik | 1/6 | 0 | Log(2/1) = 0.3 | 0.05 | 0 |
| Kakak | 0 | 1/6 | Log(2/1) = 0.3 | 0 | 0.05 |
| Pergi | 1/6 | 1/6 | Log(2/2) = 0 | 0 | 0 |
| Ke | 1/6 | 1/6 | Log(2/2) = 0 | 0 | 0 |
| Sekolah | 1/6 | 1/6 | Log(2/2) = 0 | 0 | 0 |
| Naik | 1/6 | 1/6 | Log(2/2) = 0 | 0 | 0 |
| Sepeda | 1/6 | 0 | Log(2/1) = 0.3 | 0.05 | 0 |
| Motor | 0 | 1/6 | Log(2/1) = 0.3 | 0 | 0.05 |

Dari kesimpulan perhitungan diatas menunjukan bahwa kata yang bernilai 0 tidak signifikan. Dan kata lainnya yang bernilai tidak 0 lebih signifikan. Semakin besar nilai perhitungan bobot yang diperoleh maka semakin tinggi tingkat similaritas/kesamaan dokumen .

1. **Tujuan dan Manfaat Penggunaan TF-IDF dalam Pengolahan Bahasa Alami**

* Melihat term yang penting dalam sebuah/beberapa dokumen
* Membuat kesimpulan dalam sebuah dokumen
* Mempermudah pencarian file

1. **Aplikasi Penggunaan TF-IDF**

* Pengklasifikasian berita online sesuai kategorinya.
* Pengklasifikasi mood lirik lagu.
* Pencarian dokumen

**Referensi:**

<https://informatikalogi.com/term-weighting-tf-idf/>

<https://medium.freecodecamp.org/how-to-process-textual-data-using-tf-idf-in-python-cd2bbc0a94a3>

<https://nlp.stanford.edu/IR-book/html/htmledition/maximum-tf-normalization-1.html>

<https://en.wikipedia.org/wiki/Tf%E2%80%93idf>