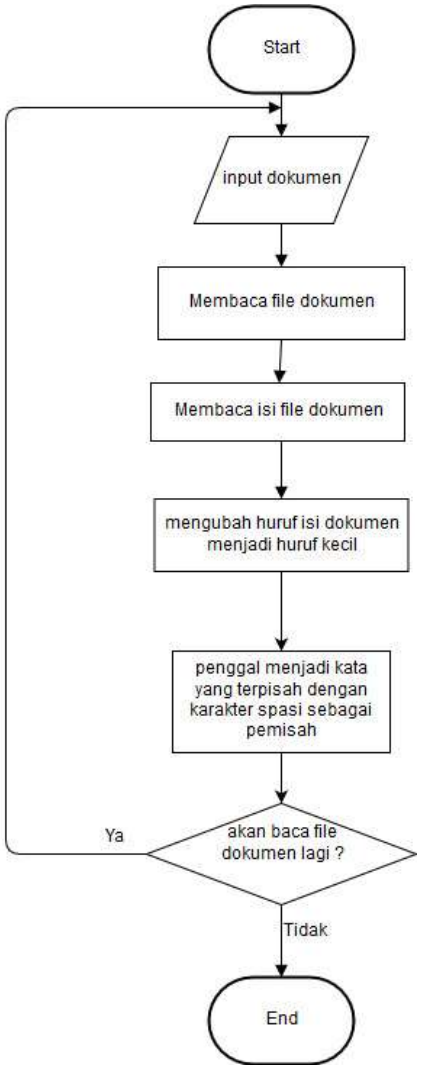


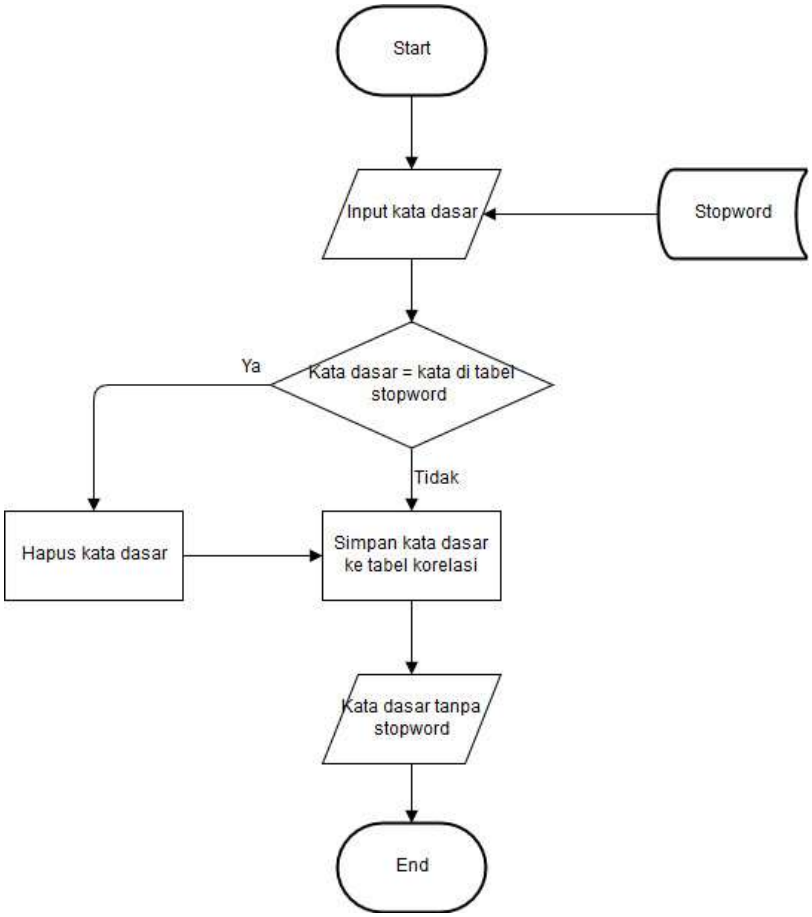
Gambar 2. 2 Tahapan proses *tokenizing*.

Pada gambar 2.2 pada proses *preprocessing* untuk tokenisasi semua *term* dalam dokumen yang dibaca diganti menjadi huruf kecil terlebih dahulu kemudian tiap *term* akan dicek apakah tanda baca atau tidak jika tanda baca maka akan dihapus. Proses akan dilanjutkan untuk membuat *term* menjadi terpisah.

c. *Stopword removal*

Tahap *stopword removal* adalah tahap mengambil kata - kata penting dari hasil token. Bisa menggunakan algoritma *stoplist* (membuang kata yang kurang penting) atau *wordlist* (menyimpan kata penting). *Stopword* adalah kata-kata

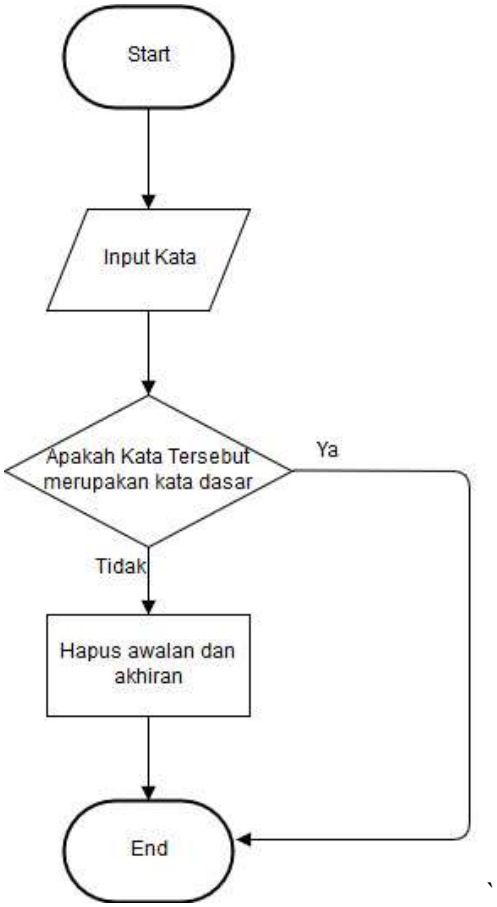
yang tidak deskriptif yang dapat dibuang dalam pendekatan *bag-of-words* [21]. Gambar 2.3 menunjukan *flowchart stopword removal*



Gambar 2. 3 Tahapan proses stopword removal atau filtering.

d. Stemming

Tahap *stemming* adalah tahap mencari *root* kata dari tiap kata hasil *filtering*. Pada tahap ini dilakukan proses pengembalian berbagai bentukan kata ke dalam suatu representasi yang sama [20]. Pencarian kata dasar dilakukan dengan menghilangkan semua imbuhan dari kata, baik itu awalan, sisipan, maupun akhiran. Gambar 2.4 Menunjukan *flowchart* dari *stemming.*

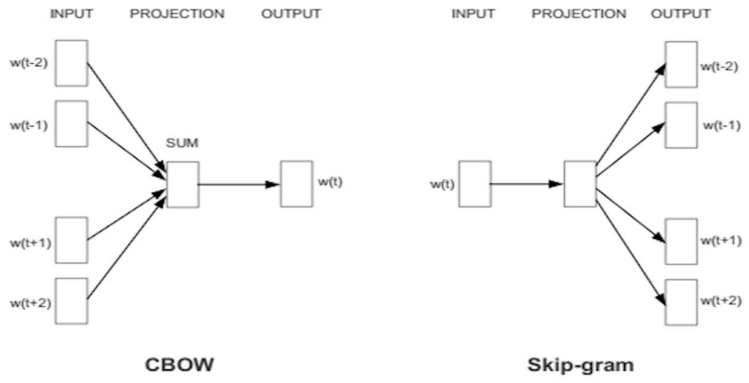


Gambar 2. 4 Tahapan proses *stemming*

2.2.5 *Word2vec*

Word2vec merupakan representasi kata dalam bentuk *vector* yang dibuat oleh Google [22]. *Word2vec* juga merupakan sekumpulan beberapa model yang saling berkaitan yang digunakan untuk menghasilkan *word embeddings*. *Word embeddings* merupakan sebutan dari seperangkat bahasa pemodelan dan teknik pembelajaran fitur *learning* pada *Natural Language Processing* (NLP) dimana setiap kata dari kosakata (*vocabulary*) memiliki vektor yang mewakili makna dari kata tersebut dan kata-kata tersebut dipetakan ke dalam bentuk vektor bilangan riil[10]. *Word2vec* menggunakan sekumpulan teks yang besar sebagai data latih (*training*) untuk membangun *vocabulary* dan menghasilkan ruang vektor yang dapat berjumlah beberapa ratus dimensi, dengan setiap kata unik pada *corpus* berupa vektor dimana

pembentukan vektor tersebut menerapkan model *Skip-gram* dan model CBOW (*Continous Bag-of-Words*) [22].



Gambar 2. 5 CBOW *Skip-gram*[10]

Model CBOW mencoba untuk memproyeksikan vektor kata-kata konteks(w_{t-1},w_{t+1}) untuk memprediksi vektor kata target $_{wt}$. sedangkan model *skip-gram* kebalikannya yaitu mencoba memprediksi vektor kata-kata yang ada dikonteks (wt-1,wt+1) diberikan vektor kata tertentu wt. Biasanya model CBOW cenderung lebih *smooth* terhadap informasi distribusional karena semua kata-kata konteks langsung diproses menjadi satu vektor sebelum akhirnya digunakan untuk memprediksi *vector* kata target oleh karna itu *corpus* yang lebih kecil ukurannya cenderung lebih baik [22]. Sedangkan model *skip-gram* membuat sepasang kata target dan konteks sebagai sebuah *instance* sehingga *skip-gram* cenderung lebih baik karena ukuran *corpus* lebih besar [22].

Tujuan dan kegunaan *Word2vec* adalah untuk mengelompokkan vektor kata-kata serupa di *vectorspace*. Artinya, ia mendeteksi persamaan secara matematik. *Word2vec* membuat vektor yang didistribusikan representasi numerik dari fitur kata, fitur seperti konteks kata-kata yang individu.

2.2.6 *Term Frequency Inverse Document Frequency* (TF-IDF)

Metode TF-IDF merupakan metode untuk menghitung bobot suatu kata (*term*) terhadap dokumen. Metode ini juga terkenal efisien, mudah dan memiliki hasil yang akurat. Metode ini menggabungkan dua konsep untuk perhitungan bobot, yaitu frekuensi kemunculan sebuah kata didalam sebuah dokumen tertentu dan inverse frekuensi dokumen yang mengandung kata tersebut [23]. Frekuensi kemunculan kata