BAB II
LANDASAN TEORI

2.1 Keluhan

Keluhan Secara definisi diartikan sebagai pernyataan atau ungkapan rasa kurang puas terhadap satu produk atau layanan jasa, baik secara lisan maupun tertulis, dari penyampai keluhan baik internal maupun eksternal. Atau sebuah ungkapan ketidakpuasan antara harapan dengan fakta terhadap apa yang diterima dalam bentuk produk maupun layanan jasa (Estrada,2011 dikutip oleh Saptono, 2016). Keluhan atau komplain berasal dari bahasa latin yang artinya adalah memukul dan ditujukan pada bagian dada seseorang. Dapat diartikan sebagai sebuah penderitaan yang mengganggu dan membuat tidak nyaman. Keluhan merupakan sebuah harapan yang belum terpenuhi (Barlow dan Moller, 1996 dikutip oleh Afriani, 2012).

Bagi orang banyak, istilah keluhan identik dengan sebuah kritik dan ancaman yang menyudutkan. Keluhan dapat bersifat *destruktif* dan bisa juga *konstruktif* bisa merusak tetapi juga bisa menjadi pemicu untuk membangun kearah yang lebih baik. Tinggal bagaimana kita menyikapi keluhan itu sendiri.

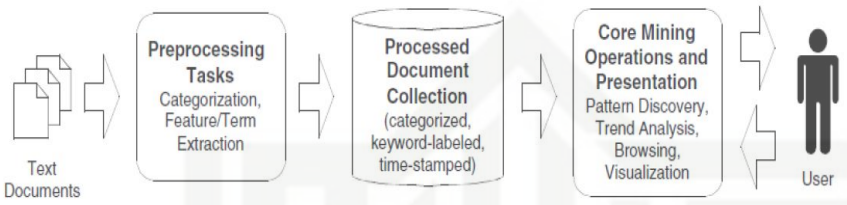
Keluhan dibedakan menjadi dua yaitu keluhan langsung dan tidak langsung. Keluhan langsung merupakan keluhan yang disampaikan secara lisan baik melalui tatap muka atau komunikasi lewat telepon. Sedangkan keluhan tidak langsung merupakan keluhan yang disampaikan secara tertulis yaitu via surat atau melalui media penyampai keluhan. Penanganan keluhan yang efektif memiliki dua kata kunci yaitu kecepatan penanganan atas keluhan dan penyelesaian keluhan.

2.2 *Text Mining*

Text mining dapat didefinisikan secara luas sebagai proses ekstraksi pola berupa informasi dan pengetahuan yang berguna dari sejumlah besar sumber datateks, seperti dokumen Word, PDF, kutipan teks, dll. *Text mining* berusaha untuk mengekstrak informasi yang berguna dari sumber data melalui

identifikasi dan eksplorasi pola yang menarik. *Text mining* banyak mengarah pada bidang penelitian data mining. Oleh karena itu, tidak mengherankan bahwa *text mining* dan data mining akan berada pada tingkat arsitektur yang sama (Feldman, dkk 2007 dikutip oleh Rosdiansyah, 2014).

Berikut gambaran sistem arsitektur *text mining* yang dicantumkan pada buku (Feldman, dkk. 2007 dikutip oleh Rosdiansyah,2014) Gambar 2.1.



Gambar 2.1 Sistem Arsitektur *Text Mining*

Text mining merupakan teknik yang digunakan untuk menangani masalah klasifikasi, *clustering*, *information extraction* dan *information retrival* (Berry dan Kogan, 2010 dikutip oleh Utomo Manalu, 2014 dikutip oleh Nurjanaty, 2016)

Pada dasarnya proses kerja dari *text mining* banyak mengapdopsi dari penelitian data mining namun yang menjadi perbedaan adalah pola yang digunakan oleh *text mining* diambil dari sekumpulan bahasa alami yang tidak terstruktur sedangkan dalam Data mining pola yang diambil dari database yang terstruktur (Han dan Kamber, 2006). *Text mining*, mengacu pada proses mengambil informasi berkualitas tinggi dari teks. Informasi berkualitas tinggi biasanya diperoleh melalui peramalan pola dan kecenderungan melalui sarana seperti pembelajaran pola statistik (Nurjanaty, 2016).

Pendekatan manual *text mining* secara intensif dalam laboratorium pertama muncul pada pertengahan 1980-an, namun kemajuan teknologi telah memungkinkan ranah tersebut untuk berkembang selama dekade terakhir. *Text mining* adalah bidang interdisipliner yang mengacu pada pencarian informasi, 30 pertambangan data, pembelajaran mesin, statistik, dan komputasi linguistik. Dikarenakan kebanyakan informasi (perkiraan umum mengatakan lebih dari 80%) saat ini disimpan sebagai teks, *text mining* diyakini memiliki potensi nilai komersial tinggi (Clara Bridge, 2011 dikutip oleh Nurjanaty, 2016)

2.3 Pembuatan *Inverted Index* dan *Text Pre-Processing*

Pembangunan *Inverted index* dari koleksi keluhan mahasiswa merupakan tugas pokok pada tahapan *pre-processing* pada *text* keluhan. Indeks dokumen keluhan adalah himpunan *term* yang menunjukkan isi yang dikandung oleh keluhan. Indeks akan membedakan suatu keluhan dari keluhan lain yang berada di dalam koleksi. Ukuran indeks yang sedikit akan memberikan hasil yang kurang maksimal sehingga beberapa *item* yang relevan terabaikan. Indeks yang banyak memungkinkan ditemukannya banyak dokumen yang relevan sekaligus dapat menaikkan jumlah dokumen yang tidak relevan, tetapi akan menurunkan kecepatan pencarian (Hyusein dkk, 2003).

Ada lima langkah dalam membentuk indeks yang dikemukakan oleh (Manning dkk, 2009):

1. Penghapusan format dan *markup* dari dalam dokumen

Pada tahap ini dilakukan penghapusan terhadap semua tag markup dan format dokumen seperti (X)HTML.

2. *Tokenization* (Pemisahan Rangkaian Kata)

Tokenization merupakan suatu proses memisahkan deretan kata menjadi potongan kata atau token. Dalam tahapan ini juga dilakukan penghilangan tanda baca dan mengubah semua token menjadi huruf kecil (*lower case*).

3. *Filtration* (Penyaringan)

Pada tahapan ini ditentukan *term* mana yang akan digunakan untuk merepresentasikan dokumen sehingga dapat mendeskripsikan isi dokumen dan membedakan dokumen tersebut dari dokumen lain di dalam koleksi. Pada tahap ini terdapat dua proses yaitu eliminasi *stopwords* dan pengambilan *wordlist*. Eliminasi *stopwords* yaitu penyaringan (*filtering*) terhadap kata-kata yang tidak memiliki arti atau tidak layak untuk dijadikan sebagai kata kunci dalam pencarian dokumen sehingga kata-kata tersebut dapat dihilangkan dari dokumen.Sedangkan *wordlist* adalah daftar kata yang mungkin digunakan sebagai kata kunci dalam pencarian dokumen. Eliminasi *stopwords* memiliki banyak keuntungan, yaitu akan mengurangi *space* pada tabel term index hingga 40% atau lebih (Baeza, 1999).

4. *Stemming* (Konversi Ke kata dasar)

Stemming adalah suatu proses yang bertujuan untuk mentransformasikan kata-kata yang terdapat dalam suatu dokumen ke kata-kata akarnya atau kata-kata dasarnya (*root word*) dengan menggunakan aturan-aturan tertentu. Sebagai contoh kata bersama, kebersamaan, menyamai, akan di *stemming* ke kata dasarnya yaitu sama. Proses *stemming* pada teks berbahasa Indonesia berbeda dengan *stemming* pada teks berbahasa inggris. Pada teks berbahasa Inggris, proses yang diperlukan hanya proses menghilangkan *sufiks*. Sedangkan pada teks berbahasa indonesia, selain *sufiks*, prefiks dan konfiks juga dihilangkan (Agusta, 2009).

5. *Indexing* (pelabelan)

Sebuah indeks selalu memetakan kembali dari setiap *term* ke dokumen dimana *term* tersebut muncul. Pengindeksan dilakukan dengan membuat *inverted index* yang terdiri dari *dictionary* dan *postings*. *Inverted index* merupakan konversi dari dokumen asli yang mengandung sekumpulan kata ke dalam daftar kata (dictionary) yang memiliki hubungan dengan dokumen terkait dimana kata-kata tersebut muncul (*postings*). *Dictionary* adalah daftar kata yang diperoleh dari hasil pengindeksan koleksi dokumen.

2.3.1 Algoritma Nazief Adriani

Algoritma Nazief dan Adriani, algoritma *stemming* untuk teks berbahasa Indonesia yang mempunyai tingkat keakuratan yang lebih baik dari algoritma lainnya (Agusta, 2009). Algoritma Nazief dan Adriani mengacu pada aturan morfologi bahasa Indonesia yang mengelompokkan imbuhan, yaitu imbuhan yang diperbolehkan atau imbuhan yang tidak diperbolehkan. Pengelompokan ini termasuk imbuhan di depan (awalan), imbuhan kata di belakang (akhiran), imbuhan kata di tengah (sisipan) dan kombinasi imbuhan pada awal dan akhir kata (konfiks) (Sahroni, R, 2012).

Berikut ini adalah langkah-langkah yang dilakukan oleh algoritma Nazief dan Adriani (Agusta, 2009) :

1. Kata yang belum di-*stemming* dicari pada kamus. Jika kata itu langsung ditemukan, berarti kata tersebut adalah kata dasar. Kata tersebut dikembalikan dan algoritma dihentikan.

2. Hilangkan *inflectional suffixes* terlebih dahulu. Jika hal ini berhasil dan *suffix* adalah partikel (“lah” atau ”kah”), langkah ini dilakukan lagi untuk menghilangkan *inflectional possessive pronoun suffixes* (“ku”, “mu” atau ”nya”).
3. *Derivational suffix* kemudian dihilangkan. Lalu langkah ini dilanjutkan lagi untuk mengecek apakah masih ada *derivational suffix* yang tersisa, jika ada maka dihilangkan. Jika tidak ada lagi maka lakukan langkah selanjutnya.
4. Kemudian *derivational prefix* dihilangkan. Lalu langkah ini dilanjutkan lagi untuk mengecek apakah masih ada *derivational prefix* yang tersisa, jika ada maka dihilangkan. Jika tidak ada lagi maka lakukan langkah selanjutnya.
5. Setelah tidak ada lagi imbuhan yang tersisa, maka algoritma ini dihentikan kemudian kata dasar tersebut dicari pada kamus, jika kata dasar tersebut ketemu berarti algoritma ini berhasil tapi jika kata dasar tersebut tidak ketemu pada kamus, maka dilakukan *recoding*.
6. Jika semua langkah telah dilakukan tetapi kata dasar tersebut tidak ditemukan pada kamus juga maka algoritma ini mengembalikan kata yang asli sebelum dilakukan *stemming*.

Kelebihan pada algoritma Nazief dan Andriani ini adalah bahwa algoritma ini memperhatikan kemungkinan adanya partikel-partikel yang mungkin mengikuti suatu kata berimbuhan. Sehingga kita dapat melihat pada rumus untuk algoritma ini yaitu adanya penempatan *possesive pronoun* dan juga partikel yang mungkin ada pada suatu kata berimbuhan. Akhir dari algoritma ini yaitu apabila pemotongan semua imbuhan telah berhasil dan hasil pemotongan imbuhan tersebut terdapat pada kamus maka algoritma ini dapat dikatakan berhasil dalam penentuan kata dasarnya. Apabila sebaliknya, bahwa setelah dilakukan pemotongan kata dan tidak terdapat pada kamus maka kata berimbuhan yang telah mengalami pemotongan dikembalikan ke keadaan semula.

Algoritma yang dibuat oleh Bobby Nazief dan Mirna Adriani ini memiliki tahap-tahap sebagai berikut :

1. Cari kata yang akan di *stemming* dalam kamus. Jika ditemukan maka diasumsikan bahwa kata tesebut adalah *root word*, maka algoritma berhenti.

2. *Inflection suffixes* (“-lah”, “-kah”, “-ku”, “-mu”, atau “-nya”) dibuang. Jika berupa partikel (“-lah”, “-kah”, “-tah” atau “-pun”) maka langkah ini diulangi lagi untuk menghapus *possesive pronouns* (“-ku”, “-mu”, atau “-nya”), jika ada.
3. Hapus *Derivation suffixes* (“-i”, “-an” atau “-kan”). Jika kata ditemukan di kamus, maka algoritma berhenti. Jika tidak maka ke langkah 3a
   a. Jika “-an” telah dihapus dan huruf terakhir dari kata tersebut adalah “-k”, maka “-k” juga ikut dihapus. Jika kata tersebut ditemukan dalam kamus maka algoritma berhenti. Jika tidak ditemukan maka lakukan langkah 3b.
   b. Akhiran yang dihapus (“-i”, “-an” atau “-kan”) dikembalikan, lanjut ke langkah 4.
4. Hapus *Derivation prefix*. Jika pada langkah 3 ada *sufiks* yang dihapus maka pergi ke langkah 4a, jika tidak pergi ke langkah 4b.
   a. Periksa kombinasi awalan-akhiran yang tidak diijinkan. Jika ditemukan maka algoritma berhenti, jika tidak pergi ke langkah 4b.
   b. For i = 1 to 3, tentukan tipe awalan kemudian hapus awalan. Jika *root word* belum juga ditemukan lakukan langkah 5. Jika sudah maka algoritma berhenti. Catatan: jika awalan kedua sama dengan awalan pertama maka algoritma berhenti.
5. Melakukan *recoding*.
6. Jika semua langkah telah selesai tetapi tidak juga berhasil maka kata awal diasumsikan sebagai *root word*. Proses selesai.

Tipe awalan ditentukan melalui langkah-langkah berikut:

a. Jika awalannya adalah: “di-”, “ke-”, atau “se-” maka tipe awalannya secara berturut-turut adalah “di-”, “ke-”, atau “se-”.
b. Jika awalannya adalah “te-”, “me-”, “be-”, atau “pe-” maka dibutuhkan sebuah proses tambahan untuk menentukantipe awalannya.
c. Jika dua karakter pertama bukan “di-”, “ke-”, “se-”, “te-”, “be-”, “me-”, atau “pe-” maka berhenti.
d. Jika tipe awalan adalah “tidak ada” maka berhenti. Jika tipe awalan adalah bukan “tidak ada” . Hapus awalan jika ditemukan.

2.3.2 *Term Weighting*

Setelah *term* di indeks selanjutnya di berikan bobot sesuai pembobotan yang dipilih, apakah pembobotan lokal, global atau kombinasi keduanya. Pembobotan lokal dilakukan dengan menghitung frekuensi kemunculan *term* dalam dokumen diekspresikan sebagai *tf* (*term frequency*). Namun, jika menggunakan pembobotan global dominasi *term* yang sering muncul dalam dokumen akan diberikan tekanan nilai, diekspresikan sebagai *idf* (*inverse document frequency*). Pembobotan Nilai *idf* ini diperlukan karena *term* yang sering muncul didalam dokumen akan di anggap sebagai *common term* (*term* umum) sehingga nilainya tidak penting, persamaan yang digunakan adalah sebagai berikut:

$$idf(t) = \log(\frac{n}{df(t)}) \quad \text{.....} \quad (2.1)$$

Keterangan:

n : Jumlah dokumen dalam dataset.

df(t) : *Document frequency* atau jumlah dokumen dalam dataset yang mengandung kata *t*.

Pada penelitian ini akan menggunakan kombinasi TF-IDF (*Term Frequncy Inverse Document Frequency*) secara bersamaan. Metode TF-IDF merupakan suatu cara untuk memberikan bobot hubungan suatu kata (*term*) terhadap dokumen (Robertson, 2005). 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.

Terdapat beberapa cara atau metode dalam melakukan pembobotan istilah pada metode TF-IDF, yaitu melalui skema pembobotan *query* dan dokumen. Pada teknik pembobotan ini, bobot istilah telah dinormalisasi. Dalam menentukan bobot suatu istilah tidak hanya berdasarkan frekuensi kemunculan istilah di satu dokumen, tetapi juga memperhatikan frekuensi terbesar pada suatu istilah yang dimiliki oleh dokumen bersangkutan. Hal ini untuk menentukan posisi relative bobot dari istilah dibanding dengan istilah-istilah lain didokumen yang sama. Selain itu teknik ini juga memperhitungkan jumlah dokumen yang mengandung istilah yang bersangkutan dan jumlah keseluruhan dokumen.

Hal ini berguna untuk mengetahui posisi relative bobot istilah bersangkutan pada suatu dokumen dibandingkan dengan dokumen-dokumen lain

yang memiliki istilah yang sama. Sehingga jika sebuah istilah mempunyai frekuensi kemunculan yang sama pada dua dokumen belum tentu mempunyai bobot yang sama.

2.4 Algoritma *K-Nearest Neighbor*

Algoritma *K-Nearest Neighbor* (K-NN) adalah sebuah metode untuk melakukan klasifikasi terhadap objek berdasarkan data pembelajaran yang jaraknya paling dekat dengan objek tersebut (Zainuddin, ddk. 2013). Rumus-rumus yang biasa digunakan sebagai ukuran jarak untuk data numerik ini antara lain:

a. *Euclidean Distance*

Ukuran ini sering digunakan dalam clustering karena sederhana.Ukuran ini memiliki masalah jika skala nilai atribut yang satu sangat besar dibandingkan nilai atribut lainnya.Oleh sebab itu, nilai-nilai atribut sering dinormalisasi sehingga berada dalam kisaran 0 dan 1.Untuk mendefinisikan jarak antara dua titik yaitu titik pada data *training* (x) dan titik pada data *testing* (y) maka digunakan persamaan *Euclidean*, seperti yang ditunjukkan pada berikut:

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(xi - yi)^2} \quad \text{.....} \quad (2.2)$$

dengan :

d : jarak antara titik pada data *training* x dan titik data *testing* y yang akan diklasifikasi, dimana x=x1,x2,…,xi dan y=y1,y2,…,yi

I : merepresentasikan nilai atribut

n : merupakan dimensi atribut.

b. *City Block Distance*

Jika tiap item digambarkan sebagai sebuah titik dalam grid, ukuran jarak ini merupakan banyak sisi yang harus dilewati suatu titik untuk mencapai titik yang lain seperti halnya dalam sebuah peta jalan.

c. *Manhattan Distance*

Manhattan Distance merupakan salah satu pengukuran yang paling banyak digunakan meliputi penggantian perbedaan kuadrat dengan menjumlahkan perbedaan absolute dari variable-variable. Fungsi ini hanya akan menjumlahkan

selisih nilai x dan y dari dua buah titik. Seperti yang ditunjukan pada persamaan berikut ini :

$$d(U,V) = \Sigma \mid Ui - Vi \mid \quad \text{.....(2.3)}$$

d. Minkwoski Metric

Ukuran ini merupakan bentuk umum dari *Euclidean Distance* dan *Manhattan Distance*.*Euclidean Distance* adalah kasus dimana nilai p=2 sedangkan *Manhattan Distance* merupakan bentuk *Minkwoski* dengan p=1. Dengan demikian, lebih banyak nilai numerik yang dapat ditempatkan pada jarak terjauh di antara 2 vektor. Seperti pada *Euclidean Distance* dan juga *Manhattan Distance*, ukuran ini memiliki masalah jika salah satu atribut dalam vektor memiliki rentang yang lebih besar dibandingkan atribut-atribut lainnya.

d. *Cosine*

Ukuran ini bagus digunakan pada data dengan tingkat kemiripan tinggi walaupun sering pula digunakan bersama pendekatan lain untuk membatasi dimensi dari permasalahan. Dalam mendefenisikan ukuran jarak antar k yang digunakan beberapa algoritma untuk menentukan k mana yang terdekat. Pada penelitian ini pengukuran jarak yang digunakan adalah *Cosine Similarity* rumus yang digunakan adalah sebagai berikut:

$$Cos(i,k) = \frac{\sum_k (d_1 d_k)}{\sqrt{\sum_k d_{ik}^2}\sqrt{\sum_k d_{jk}^2}} \quad \text{.....(2.4)}$$

Keterangan:

$\sum_k (d_1 d_k)$: vector dot produk dari i, dan k

$\sqrt{\sum_k d_{ik}^2}$: Panjang vector i

$\sqrt{\sum_k d_{jk}^2}$: Panjang vector k

Algoritma *K-Nearest Neighbor* (Krisandi, dkk. 2013) adalah algoritma yang menentukan nilai jarak pada pengujian data *testing* dengan data *training* berdasarkan nilai terkecil dari nilai ketetanggaan terdekat didefinisikan sebagai berikut:

$$Dnn\ (C_1 C_2) = min_{1 \le i \le r, 1 \le j \le s} d(y_1\ z_1) \quad \text{..... (2.5)}$$

K-Nearest Neighbor adalah algoritma untuk mengklasifikasi objek baru berdasarkan atribut dan *training samples* (data latih). Dimana hasil dari sampel uji yang baru diklasifikasikan berdasarkan mayoritas dari kategori pada *K-Nearest Neighbor*. Algoritma *K-Nearest Neighbor* menggunakan klasifikasi ketetanggaan sebagai nilai prediksi dari sampel uji yang baru (Krisandi, dkk. 2013). Data latih akan dibangun dengan memperhatikan keseimbangan dokumen satu sama lain. Adapun algortima *K-Nearest Neighbor* dapat dijelaskan dengan keterangan berikut :

1. Hitung jarak antara data sampel (data uji) dengan data latih yang telah dibangun. Salah satu persamaan dalam menghitung jarak kedekatan dapat menggunakan persamaan (2.4) *Cosine Similarity*
2. Menentukan parameter nilai *k* = jumlah tetanggaan terdekat.
3. Mengurutkan jarak terkecil dari data sample.
4. Pasangkan kategori sesuai dengan kesesuaian.
5. Cari jumlah terbanyak dari tetanggaan terdekat dengan persamaan (2.5). Kemudian tetapkan kategori.

2.5 Evaluasi

Sebuah sistem yang melakukan klasifikasi diharapkan dapat melakukan klasifikasi semua set data dengan benar. Akan tetapi, tidak dapat dipungkiri suatu sistem tidak bisa bekerja 100% benar. Oleh karena itu sebuah sistem klasifikasi harus di ukur kinerjanya. Umumnya cara mengukur kinerja klasifikasi menggunakan *Confusion matrix*.

Confusion matrix merupakan tabel yang mencatat hasil kerja klasifikasi. *True positives* (TP) dan *True negatives* (TP) memberikan informasi ketika *classifier* benar, sedangkan *False positives* (FP) dan *False Negatives* (FP) memberikan informasi ketika *classifier* salah (Prasetyo, 2014). Tabel 2.1 adalah contoh dari *confusion matrix*.

Tabel 2.1 *Confusion Matrix* (Prasetyo, 2014)

		Kelas hasil prediksi	
		Positif	**Negatif**
Kelas asli	**Positif**	*True positives* (TP)	*False positives*(FP) Error tipe II
	Negatif	*False Negatives*(FP) Error tipe I	*True negatives*(TP)

Akurasi merupakan persentase dari data yang diprediksi secara benar. Perhitungan akurasi adalah :

$$Akurasi = \frac{(TP+TN)}{(TP+TN+FP+FN)} \quad \text{.....(2.6)}$$

Keterangan :

TP : *True positives*, merupakan jumlah data dengan kelas positif yang diklasifikasikan secara benar sebagai positif.

TN : *True negatives*, merupakan jumlah data dengan kelas negatif yang diklasifikasikan secara benar sebagai negatif.

FP : *False positives*, merupakan jumlah data dengan kelas positif diklasifikasikan secara salah sebagai negatif.

FN : *False negatives*, merupakan jumlah data dengan kelas negatif diklasifikasikan secara salah sebagai positif.

2.6 Penelitian Terkait

Adapun penelitian terkait penelitian ini akan dijelaskan pada Tabel berikut:

Tabel 2.2 Penelitian Terkait Penerapan Metode *K-Nearest Neighbor* untuk Klasifikasi Terhadap Keluhan

Nama	**Judul Penelitian**	**Kesimpulan**	**Tahun**
Aisha A.M, Ristu Saptono, Rini A.	Sistem Klasifikasi *Feedback* Pelanggan dan Rekomendasi Solusi Atas Keluhan di UPT Puskom UNS dengan Algoritma Naïve Bayes *Classifier* dan *Cosine Similarity*.	Proses klasifikasi dengan algoritma Naïve Bayes *Classifier* untuk proses pelatihan memiliki tingkat akurasi terendah 86.67% dengan data pelatihan sebanyak 30 *mentions* dan tingkat akurasi tertinggi 100% dengan data pelatihan sebanyak 20	2016

Nama	Judul Penelitian	Kesimpulan	Tahun
		mentions.	
Vijayarani, Ilamathi J,Nithya	*Preprocessing Techniques for Text Mining - An Overview*	Text mining adalah proses pencarian atau penggalian informasi yang berguna dari data tekstual. Pada penelitian ini menggunakan teknik eliminasi *stopword* dan *stemming* pada *Pre-prosessing*	2016
Aisha A.M, Ristu Saptono, Rini A.	Sistem Klasifikasi *Feedback* Pelanggan dan Rekomendasi Solusi Atas Keluhan di UPT Puskom UNS dengan Algoritma Naïve Bayes *Classifier* dan *Cosine Similarity*	Proses klasifikasi dengan algoritma Naïve Bayes *Classifier* untuk proses pelatihan memiliki tingkat akurasi terendah 86.67% dengan data pelatihan sebanyak 30 *mentions* dan tingkat akurasi tertinggi 100% dengan data pelatihan sebanyak 20 *mentions*.	2015
Widyastuti	Sistem Klasifikasi Dokumen Bahasa Jawa dengan Metode *K-Nearest Neighbor (K-NN)*	Pada penelitian ini klasifikasi dokumen berdasarkan empat kategori yaitu politik, ekonomi, kesehatan dan pendidikan. Didapat hasil pengujian akurasi tertinggi adalah 95% , sedangkan akurasi terendah mencapai 92%.	2014
Kanojia dan Motnawani	*Comparison of Naive Basian and K-Nearest Neighbor Classifier*	Bila kedua metode klasifikasi ini diterapkan pada data yang sama untuk mengetahui hasil optimal menunjukkan bahwa metode klasifikasi *K-Nearest Neighbor* memberikan akurasi yang lebih tinggi yaitu 83,65% sedangkan *Naive bayes* 75,77%	2013