Penerapan Pohon Untuk Klasifikasi Dokumen Teks Berbahasa Inggris

Riza Ramadan – NIM : 13503037

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
Jl. Ganesha 10, Bandung
E-mail : if13037@students.if.itb.ac.id

Abstrak

Klasifikasi dokumen teks sebenarnya adalah permasalahan yang mendasar dan penting. Didalam dokumen teks, tulisan yang terkandung adalah bahasa alami manusia, yang merupakan bahasa dengan struktur yang kompleks dan jumlah kata yang sangat banyak. Oleh karena itu, permasalahan ini sangat menantang dikarenakan penggunaan bahasa alami tersebut.

Setelah melewati beberapa penelitian, ternyata dengan mengesampingkan pemrosesan bahasa natural dan menggunakan pendekatan statistik yaitu dengan hanya menganggap dokumen tersebut merupakan *bag of word* didapat performa yang cukup memuaskan. Oleh karena itu, dikembangkannya berbagai metode klasifikasi yang tidak memperhitungkan semantik dari dokumen tersebut.

Didalam makalah ini, dibahas metode klasifikasi dokumen teks dengan menggunakan konsep pohon, yaitu dengan memperhitungkan jumlah kemunculan semua kata yang muncul dalam dokumen teks tersebut. Sebagai pembanding untuk metode tersebut, dibahas juga metode seperti metode *Naive Bayes*, yaitu metode dengan menghitung probabilitas kemunculan kata, metode *K-Nearest Neighbor* (KNN), yaitu metode yang memperhitungkan kemiripan jumlah kemunculan kata antara satu dokumen dengan dokumen lain, dan metode *Artificial Neural Network* (ANN), yaitu metode yang membangun model berdasarkan jaringan saraf manusia untuk permasalahan yang diberikan. Setelah membahas metode-metode yang ada, dilakukan percobaan terhadap data untuk mengetahui performa masing-masing metode untuk diketahui sebenarnya metode mana yang paling baik untuk digunakan dalam permasalahan klasifikasi teks.

Kata Kunci : Text document classification, *Artificial Neural Network, K-Nearest Neighbor, Naive Bayes,* Weka, YALE, *Natural Language Processing.*

1. Pendahuluan

1.1. Latar Belakang

Dalam 20 tahun terakhir, jumlah dokumen teks dalam bentuk digital telah berkembang sangat besar dari segi ukuran. Sebagai konsekuensi, sangatlah penting untuk bisa menorganisir dan mengklasifikasi dokumen secara otomatis. Penelitian pada pengklasifikasian teks bertujuan untuk mempartisi himpunan dokumen yang tidak terstruktur ke dalam kelompok-kelompok yang menggambarkan isi dari dokumen. Ada dua varian utama dalam pengklasifikasian teks: *clustering* teks dan pengkategorian teks. *Clustering* teks berhubungan dengan menemukan sebuah struktur kelompok yang belum kelihatan dari sekumpulan dokumen. Sedangkan pengklasifikasian teks dapat dianggap sebagai *task* untuk membentuk struktur dari penyimpanan dokumen berdasarkan pada struktur kelompok yang sudah diketahui sebelumnya.

Dengan semakin meningkatnya kebutuhan untuk pengklasifikasian dokumen, pencarian akan algoritma untuk membantu melakukan aktivitas tersebut juga semakin dikembangkan. Terdapat beberapa metode yang ada untuk pengklasifikasian dokumen yang muncul dan digunakan dalam aplikasi yang berbeda-beda. Beberapa metode pengklasifikasian dokumen yang cukup sering digunakan adalah Pohon, Naïve Bayes Classifier, K-Nearest Neighbor dan Neural Network.

Pengklasifikasian dokumen muncul dalam berbagai aplikasi, meliputi e-mail *filtering*, *mail routing*, *spam filtering*, pengontrolan berita, pengindeksan otomatis pada artikel ilmiah, indetifikasi terhadap genre dokumen, authorship attribution, survey pengkodean dan lain-lain. Pengkategorian teks secara otomatis sangatlah menarik karena mengorganisir dokumen teks secara manual akan menjadi sangat mahal, atau tidak *feasible* karena keterbatasan waktu dari aplikasi atau jumlah dokumen yang digunakan. Selain itu, meski performa klasifikasi dengan menggunakan tenaga manusia menunjukkan hasil yang baik, namun tidak sempurna. Tingkat akurasinya hanya berupa 80% - 95% saja. Oleh karena itu, kebutuhan akan klasifikasi otomatis semakin diperlukan.

1.2. Motivasi

Dengan adanya latar belakang di atas, pengklasifikasian dokumen adalah suatu hal yang penting dan kebutuhan terhadapnya akan semakin meningkat seiring dengan berjalannya waktu, karena dokumen semakin lama akan semakin banyak dan ukuran *harddisk* aka semakin besar. Sehingga perlu dilakukan pengkajian metode untuk klasifikasi dokumen teks dan uji coba terhadap hal tersebut melalui melakukan eksperimen terhadap beberapa metode-metode klasifikasi yang dikaji tersebut, yaitu metode dengan menggunakan pohon, *naïve Bayes*, *K-Nearest Neighbor* dan *Neural Network*. Eksperimen tersebut dilakukan menggunakan aplikasi klasifikasi yang sudah ada, yaitu weka dan YALE.

Eksperimen dilakukan untuk mengklasifikasi dokumen berita (*news document*) dengan tujuan untuk melakukan pengkategorian dokumen berdasarkan jenis berita, misalnya saja apakah dokumen A merupakan dokumen yang berisi berita olahraga atau berisi berita politik. Data set yang digunakan adalah data yang didapatkan dari http://www.cs.cmu.edu/afs/cs/project/theo-11/www/naive-bayes.html, yaitu data 20 *news group* yang terdiri dari 20.000 dokumen dengan 20 jenis berita.

2. Permasalahan

Tentunya mudah untuk melakukan klasifikasi manual (dengan tenaga manusia) dokumen teks dengan jumlah yang kecil, misalkan klasifikasi 100 dokumen teks. Namun akan menjadi suatu hal tidak memungkinan untuk dilakukan bila jumlah dokumen teks tersebut berjumlah puluhan ribu. Akan sangat memakan waktu dan biaya.

Karena dokumen teks adalah klasifikasi dokumen yang berupa kata-kata pasti memiliki maksud tertentu yang disampaikan dalam keterurutan kata-kata tersebut, dengan mengelompokkan teks berdasarkan maksud yang terkandung, logikanya akan didapat suatu metode klasifikasi yang memberikan performa yang tinggi, karena menang dokumen – dokumen tersebut diklasifikasikan berdasarkan kategori tertentu. Dengan latar belakang seperti ini, dikembangkannya pemrosesan bahasa alami.

Pemrosesan bahasa alami diasumsikan sebagai proses yang merentang dari penerimaan data suara yang berupa perintah dari pengguna sampai mengeksekusi perintah yang dimaksudkan tersebut. Dengan asumsi seperti ini, pemrosesan bahasa alami terdiri dari beberapa tahap, yaitu :

1. Pengenalan data suara perintah
   Tahap ini adalah proses membangkitkan kalimat dari data suara yang dimasukkan oleh pengguna. Intrepretasi data ini biasanya berupa grafik *sound audio*, sehingga perhitungan rumus matematika yang rumit diperlukan dalam tahap ini. Permasalahan yang muncul dalam tahap ini antara lain perbedaan grafik yang dihasilkan oleh orang yang sama meski mengucapkan kalimat yang sama namun dalam waktu yang berbeda, perbedaan grafik yang dihasilkan oleh orang yang berbeda meski mengucapkan kalimat yang sama, gangguan suara - suara lingkungan yang ada, dan perbedaaan frekuensi suara yang signifikan antara pria dan wanita. Permasalahan - permasalahan tersebut adalah permasalahan yang sangat rumit sehingga memerlukan komputasi yang mahal dan algoritma yang canggih untuk menyelesaikannya.

2. Pembentukan struktur kalimat perintah
   Asumsi tahap sebelumnya berjalan dengan sempurma, yaitu menghasilkan urutan kata yang sama dengan yang diucapkan oleh pengguna, perlu dilakukan penstrukturan terhadap kalimat perintah tersebut, karena kalimat yang diterima tidak dapat dianggap sebagai *array of word* jika ingin didapat arti yang terkandung didalamnya, karena mustahil untuk dilakukan. Permasalahan yang muncul pada tahap ini adalah pembentukan struktur itu sendiri. Suatu kata dalam kalimat tidak hanya memiliki satu

posisi pada pola, misalkan suatu kata hanya bisa menjadi subjek saja. Satu kata saja dapat berupa subjek, objek atau lainnya meski memiliki keterurutan karakter yang sama.

3. Pengartian kalimat perintah
Tahap yang sulit, yaitu mengartikan maksud dari kalimat tersebut. Permasalahannya antara lain adalah bagaimana membangkitkan arti kalimat tersebut tanpa adanya informasi sebelumnya, dan ambiguitas. Apabila dibangun suatu basis data yang berisikan seluruh arti dari suatu kata, harus dibangun juga basis data yang berisikan arti kata dalam konteks seluruh kalimat yang mungkin. Bila sudah dibangun basis data tersebut, permasalahan ambiguitas muncul ke permukaan. Ambiguitas ini saja bagi manusia yang menciptakan bahasa alami masih merupakan suatu masalah, apalagi komputer yang "tidak cerdas", akan sulit untuk membedakan arti yang mana yang dimaksud oleh pengguna

4. Eksekusi perintah yang dimaksud
Tidak seperti tahap - tahap sebelumnya, tahap ini relatif mudah karena hanya perlu menyamakan perintah yang diterima dengan daftar perintah yang ada untuk kemudian dieksekusi. Namun untuk sampai tahap ini, tahap - tahap sebelumnya perlu dilakukan dengan sempurna.

Melihat begitu rumit permasalahan pemrosesan bahasa alami ini, perlu dibuat metode lain yang lebih sederhana dan *feasible* untuk dilakukan. Oleh karena itu, klasifikasi yang berdasarkan arti yang terkandung tidak dilakukan, dan sebagai gantinya, dikembangkan metode yang hanya memperhitungkan jumlah kemunculan kata dalam dokumen teks tersebut.

3. Metode Klasifikasi dengan Pohon

Metode pohon, yang selanjutnya disebut dengan, *Iterative Dichotomiser* 3 (ID3) merupakan sebuah metode yang digunakan untuk membangkitkan pohon keputusan. Algoritma pada metode ini berbasis pada Occam's razor: lebih memilih pohon keputusan yang lebih kecil (teori sederhana) dibanding yang lebih besar. Tetapi tidak dapat selalu menghasilkan pohon keputusan yang paling kecil dan karena itu occam's razor bersifat heuristik. Occam's razor diformalisasi menggunakan konsep dari entropi informasi.

Secara ringkas, cara kerja Algoritma ID3 dapat digambarkan sebagai berikut:

1. Ambil semua atribut yang tidak terpakai dan hitung entropinya yang berhubungan dengan test sample.

2. Pilih atribut dimana nilai entropinya minimum

3. Buat simpul yang berisi atribut tersebut

Adapun *sample data* yang digunakan oleh ID3 memiliki beberapa syarat, yaitu:

1. Deskripsi atribut-nilai. Atribut yang sama harus mendeskripsikan tiap contoh dan memiliki jumlah nilai yang sudah ditentukan.

2. Kelas yang sudah didefinisikan sebelumnya. Suatu atribut contoh harus sudah didefinisikan, karena mereka tidak dipelajari oleh ID3.

3. Kelas-kelas yang diskrit. Kelas harus digambarkan dengan jelas. Kelas yang kontinu dipecah-pecah menjadi kategori-kategori yang relatif, misalnya saja metal dikategorikan menjadi "*hard*, *quite hard*, *flexible*, *soft*, *quite soft*".

4. Jumlah contoh (*example*) yang cukup. Karena pembangkitan induktif digunakan, maka dibutuhkan *test case* yang cukup untuk membedakan pola yang valid dari peluang suatu kejadian.

Pemillihan atribut pada ID3 dilakukan dengan properti statistik, yang disebut dengan information gain. *Gain* mengukur seberapa baik suatu atribut memisahkan *training example* ke dalam kelas target. Atribut dengan informasi tertinggi akan dipilih. Dengan tujuan untuk mendefinisikan *gain*, pertama-tama digunakanlah ide dari teori informasi yang disebut entropi. Entropi mengukur jumlah dari informasi yang ada pada atribut.

Rumus menghitung entropi informasi adalah:

$$\text{Entropy}(S) \equiv -p_+ \log_2 p_+ - p_- \log_2 p_-$$

Dan rumus untuk menghitung *gain* adalah:

$$\text{Gain}(S,A) \equiv \text{Entropy}(S) - \sum \frac{|Sv|}{|S|} \text{Entropy}(Sv)$$

4. Aplikasi Metode Pohon terhadap Permasalahan Klasifikasi

Pada seluruh dokumen teks yang ada, setiap kata yang muncul kecuali kata umum bahasa inggris (misalnya *at, on, a, an, the*) dijadikan atribut untuk perhitungan entropi dan *gain*. Setiap dokumen dijadikan *instance* yang memiliki nilai berdasarkan atribut yang ada. Nilai dari atribut tersebut adalah biner (0 dan 1), dimana nilai 0 mengartikan tidak muncul kata tersebut pada sutu dokumen, dan 1 mengartikan bahwa kata tersebut tedapat dalam dokumen yang berkaitan. Dengan asumsi seperti ini, dibangun pohon sebagai model dimana klasifikasi dilakukan terhadap model tersebut.

Pembangunan model ini dilakukan dengan cara menghitung nilai *gain* seluruh atribut yang ada, untuk kemudian dipilih atribut yang memiliki nilai *gain* tertinggi sebagai akar dari pohon yang dibentuk. Kemudian dari akar tersebut, simpul anak yang dibangkitkan hanya dua, kiri dan kanan, hal ini dikarenakan nilai dari suatu atribut hanya dua, sehingga pohon yang terbentuk adalah pohon biner. Dan untuk setiap simpul anak, dilakukan proses yang sama dengan proses sebelumnya, namun tidak lagi menyertakan simpul-simpul yang berada diatasnya. Dengan cara seperti ini, algoritma yang digunakan adalah algoritma rekursif.

5. Metode Klasifikasi dengan *Naive Bayes*

Naïve bayes classifier adalah suatu *classifier* probabilitas sederhana yang didasarkan pada pengaplikasian teorema Bayes dengan asumasi yang kuat (*naïve*) dan bebas (*independence*).

Bergantung pada sifat dasar dari model probabilitas, *naïve Bayes classifier* dapat dilatih dengan sangat efisien pada kondisi *supervised learning*. Pada banyak aplikasi praktikal, perkiraan parameter untuk model *naïve Bayes* menggunakan metode *maximum likelihood* (ketetanggan maksimum), sehingga *naïve Bayes* dapat digunakan tanpa perlu kepercayaan pada probabilitas Bayesian atau tanpa menggunakan metode Bayesian sama sekali. Walaupun rancangan dari *naïve Bayes classifier* bersifat *naïve* dan asumsinya terlalu disederhanakan, *naïve Bayes classifier* biasanya bekerja lebih baik dari yang diharapkan pada situasi dunia nyata yang kompleks.

Secara abstrak, model probabilitas untuk classifier adalah model kondisional:

$$P\ (C, F_1, F_2, \ldots\ldots\ldots, F_n)$$

pada variabel kelas dependen C dengan jumlah hasil atau kelas yang kecil, kondisional pada beberapa variabel fitur F_1 sampai F_n. Masalah yang dihadapi adlah apabila jumlah dari fitur n besar atau ketika fitur tersebut dapat menangani nilai dengan jumlah yang sangat banyak, maka tidak mungkin mendasari model tersebut dengan tabel probabilitas.
Naïve bayes classifier dapat dikombinasikan dengan model probabilitas dengan *decision rule*. Salah satu aturan yang umum digunakan adalah mengambil hipotesis yang paling mungkin; hal ini disebut dengan *maximun a posteriori* atau *MAP decision rule*. *Classifier* tersbut adalah fungsi klasifikasi yang didefinisikan sebagai berikut:

$$\text{classify}\ (f_1, \ldots\ldots\ldots, f_n)$$
$$= \text{argmax}_c\ P\ (C = c)\ \prod_{1}^{n}\ p(F1 = f1 \mid C = c)$$

Salah satu penggunaan *naïve Bayes classifier* yang paling sering ditemui adalah pengklasifikasian dokumen. Contoh masalah yang sering diangkat adalah pengklasifikasian dokumen berdasarkan isinya, misalnya memisahkan antara e-mail *spam* dan bukan *spam*. Dokumen-dokumen tersebut diambil dari sejumlah kelas pada dokumen dimana dapat dimodelkan sebagai sekumpulan kata dimana probabilitas bahwa kata ke-i dari dokumen yang diberikan muncul pada dokumen dari kelas C dapat dituliskan sebagai berikut:

$$P\ (w_i \mid C)$$

6. Aplikasi Metode *Naive Bayes* terhadap Permasalahan Klasifikasi

Seperti halnya pada metode Pohon, pada *Naive Bayes*, kata-kata tersebut juga dianggap sebagai atribut untuk dihitung probabilitasnya berdasarkan nilai biner. Bedanya adalah pada *Naive Bayes*, probabilitas dihitung berdasarkan seluruh dokumen yang ada, karena diperlukan informasi berdasarkan seluruh dokumen.

Kemudian setelah didapat nilai masing – masingnya, dihitung kemungkinan untuk masuk ke dalam kategori yang mana, perhitungan dilakukan sejumlah jenis kategori yang ada, dalam kasus ini 20 kali perhitungan. Nilai terbesar itu yang dianggap sebagai kategori yang tepat untuk dokumen tersebut.

7. Metode Klasifikasi dengan *K-Nearest Neigbhor*

K-Nearest Neighbor (KNN) adalah suatu metode yang menggunakan algoritma *supervised* dimana hasil dari *query instance* yang baru diklasifikasikan berdasarkan mayoritas dari kategori pada KNN. Tujuan dari algoritma ini adalah mengklasifikasi objek baru berdasakan atribut dan *training sample*. *Classifier* tidak menggunakan model apapun untuk dicocokkan dan hanya berdasarkan pada memori. Diberikan titik *query*, akan ditemukan sejumlah K objek atau (titik *training*) yang paling dekat dengan titik *query*. Klasifikasi menggunakan *voting* terbanyak di antara klasifikasi dari K objek. Algoritma KNN menggunakan klasifikasi ketetanggaan sebagai nilai prediksi dari *query instance* yang baru.

Algoritma metode KNN sangatlah sederhana, bekerja dengan berdasarkan pada jarak terpendek dari *query instance* ke *training sample* untuk menentukan KNN nya. Setelah mengumpulkan KNN, kemudian diambil mayoritas dari KNN untuk dijadikan prediksi dari *query instance*.

Data untuk algoritma KNN terdiri dari beberapa atribut *multi-variate* X_i yang akan digunakan untuk mengklasifikasikan Y. Data dari KNN dapat dalam skala ukuran apapun, dari ordinal ke nominal.

KNN memiliki beberapa kelebihan yaitu bahwa dia tangguh terhadap training data yang *noisy* dan efektif apabila *training data*-nya besar. Sedangkan kelemahan dari KNN adalah KNN perlu menentukan nilai dari parameter K (jumlah dari tetangga terdekat), pembelajaran berdasarkan jarak tidak jelas mengenai jenis jarak apa yang harus digunakan dan atribut mana yang harus digunakan untuk mendapatkan hasil yang terbaik, dan biaya komputasi cukup tinggi karena diperlukan perhitungan jarak dari tiap *query instance* pada keseluruhan *training sample.*

8. Aplikasi Metode *K-Nearest Neigbhor* terhadap Permasalahan Klasifikasi

Pada kasus ini, perhitungan jarak terdekat yang diperlukan dalam algoritma metode *K-Nearest Neigbhor* ini adalah jumlah kemiripan dihitung dari kemiripan kemunculan kata yang dimiliki suatu dokumen. Nantinya dari sejumlah *instance* yang memenuhi *threshold* tertentu, kategori terbanyak yang munculkan yang akan menjadi ketegori *instance* yang diuji

9. Metode Klasifikasi dengan *Artifical Neural Network*

Artificial Neural Network (AANN) merupakan suatu *tool* pemodelan data yang sangat hebat yang mampu untuk menangkap dan merepresentasikan hubungan *input/output* yang kompleks. Motivasi pengembangan terknologi ANN dimulai dari keinginan untuk mengembangkan suatu sistem buatan yang mampu melakukan pekerjaan “intelligent” seperti yang dilakukan oleh otak manusia. ANN menyerupai otak manusia dengan dua cara seperti berikut:

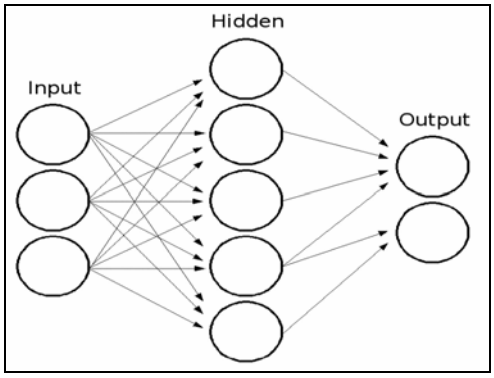
1. ANN mendapatkan pengetahuan melalui pembelajaran.

2. Pengetahuan pada ANN disimpan dalam suatu kekuatan hubungan inter-neuron yang dikenal sebagai bobot synaptic.

Kekuatan dan keuntungan yang utama dari ANN terletak pada kemampuannya untuk mereprentasikan hubungan baik secara linear maupun non-linear dan kemampuan mereka untuk mempelajari hubungan tersebut secara langsung pada data yang sedang dimodelkan.

Model ANN yang paling umum adalah *multilayer perceptron* (MLP). Tipe ANN ini dikenal sebagai jaringan *supervised* karena membutuhkan *output* yang diinginkan untuk pembelajaran. Tujuan dari tipe jaringan ini adalah untuk membuat sebuah model yang dengan tepat memetakan *input* ke *output* dengan menggunakan data historis sehingga model tersebut kemudian dapat digunakan untuk memproduksi *output* ketika *output* yang diharapkan tidak diketahui.

Selain itu, tipe paling umum dari ANN juga dapat terdiri dari 3 grup atau layer dari unit: layer dari ”input” unit dihubungkan dengan layer dari “hidden” unit, dimana kemudian dihubungkan ke layer dari “output“ unit (dapat dilihat pada gambar berikut).



MLP dan banyak jenis ANN lainnya melakukan pembelajaran menggunakan algoritma yang dinamakan *backpropagation*. Dengan *backpropragation*, data *input* secara berulang-ulang ditampilkan pada ANN. Dengan setiap presentasi, *output* dari ANN dibandingkan ke *output* lain yang diinginkan dan error pun dihitung. Error ini kemudian di berikan kembali (backpropagated) ke ANN dan digunakan untuk mengatur bobot ketika error mengalami penurunan pada setiap iterasi dan model ANN semakin mendekati untuk menghasilkan *output* yang diinginkan. Proses ini dinamakan "training".

Karena ANN sangat baik dalam mengidentifikasi pola atau tren dari data, ANN sangat cocok untuk kebutuhan dalam melakukan prediksi atau perkiraan, meliputi:

1. Perkiraan penjualan
2. Kontrol proses pada industri
3. Penelitian pelanggan
4. Validasi data
5. Manajemen resiko
6. Target pemasaran

10. Aplikasi Metode *Artifical Neural Network* terhadap Permasalahan Klasifikasi

Pembentukan model ANN untuk permasalahan ini agak rumit, karena algoritma ANN memiliki banyak parameter yang dapat di atur-atur oleh pengguna, dan perbedaan parameter itu sangat berpengaruh pada waktu proses, tingkat akurasi, dan jumlah *node* yang dibangun. Proses pembangunan model ANN begitu rumit sehingga memerlukan waktu yang lama untuk membangunnya dari data *training* yang disediakan, namun penentuan kategori data *test* sangat cepat.

11. Prosedur Uji Coba

11.1. Preprocessing

Langkah-langkah preprocessing yang diterapkan adalah sebagai berikut:

Dari seluruh kata yang ada dalam 20.000 dokumen di 20_newsgroups dimasukkan ke dalam basis data beserta jumlah kemunculan katanya. Kata-kata yang dimasukkan ke dalam basis data tidak termasuk kata dengan panjang < 2 karakter, header dokumen, angka, tanda baca, dan karakter-karakter selain huruf alfabet. Sebagai gambaran, karakter yang berwarna merah tidak dimasukkan ke dalam basis data (Gambar 1)

Setelah 20.000 dokumen sudah dimasukkan ke dalam basis data, dilakukan penghilangan stop-word seperti: about, above, across, after, afterwards, again, dan lain sebagainya.

Dari daftar kata yang sudah 'bersih' tersebut dipilih 2.000 kata dengan jumlah terbanyak sebagai atribut untuk eksperimen naive bayes, k-nearest neighbor, dan ID3 (untuk eksperimen artificial neural network dipilih 100 kata dengan jumlah terbanyak).

Setelah itu, 20.000 dokumen kemudian dipartisi menjadi 10 partisi, dengan masing-masing bagian terdiri dari 1.300 dokumen training dan 700 dokumen test.

Dari masing-masing partisi diciptakan file input untuk eksperimen berdasarkan 2.000 (atau 100) kata yang dihasilkan pada langkah nomor 3. Contoh file input adalah sebagai berikut:

```
@relation newsgroup

@attribute windows {1,0}
@attribute cars {1,0}
@attribute think {1,0}
@attribute new {1,0}
@attribute time {1,0}
@attribute year {1,0}
@attribute use {1,0}
@attribute people {1,0}
…
@attribute walk {1,0}
@attribute ease {1,0}
@attribute GS300 {1,0}
@attribute Cutting {1,0}
@attribute seals {1,0}
@attribute designation {1,0}
@attribute equipped {1,0}
@attribute kategori
{1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20}

@data
0,0,0,1,0,1,0,1, … ,0,1,0,0,0,0,0,0,1,0,0,0,0,1,5
0,0,0,1,0,1,0,1, … ,0,1,0,0,0,0,0,0,1,0,0,0,0,1,4
0,0,0,1,0,1,0,1, … ,0,1,0,0,0,0,0,0,1,0,0,0,0,1,9
0,0,0,1,0,1,0,1, … ,0,1,0,0,0,0,0,0,1,0,0,0,0,1,10
0,0,0,1,0,1,0,1, … ,0,1,0,0,0,0,0,0,1,0,0,0,0,1,5
…
0,0,0,1,0,1,0,1, … ,0,1,0,0,0,0,0,0,1,0,0,0,0,1,3
0,0,0,1,0,1,0,1, … ,0,1,0,0,0,0,0,0,1,0,0,0,0,1,2
0,0,0,1,0,1,0,1, … ,0,1,0,0,0,0,0,0,1,0,0,0,0,1,1
0,0,0,1,0,1,0,1, … ,0,1,0,0,0,0,0,0,1,0,0,0,0,1,2
0,0,0,1,0,1,0,1, … ,0,1,0,0,0,0,0,0,1,0,0,0,0,1,15
0,0,0,1,0,1,0,1, … ,0,1,0,0,0,0,0,0,1,0,0,0,0,1,9
0,0,0,1,0,1,0,1, … ,0,1,0,0,0,0,0,0,1,0,0,0,0,1,14
0,0,0,1,0,1,0,1, … ,0,1,0,0,0,0,0,0,1,0,0,0,0,1,5
0,0,0,1,0,1,0,1, … ,0,1,0,0,0,0,0,0,1,0,0,0,0,1,12
0,0,0,1,0,1,0,1, … ,0,1,0,0,0,0,0,0,1,0,0,0,0,1,7
0,0,0,1,0,1,0,1, … ,0,1,0,0,0,0,0,0,1,0,0,0,0,1,20
```

Path: cantaloupe.srv.cs.cmu.edu!magnesium.club.cc.cmu.edu!pitt.edu!zaphod.mps.ohio-state.edu!swrinde!network.ucsd.edu!mvb.saic.com!zippy.telcom.arizona.edu!bpavms.bpa.arizona.edu!dmittleman
From: dmittleman@bpavms.bpa.arizona.edu (Daniel Mittleman)
Newsgroups: comp.os.ms-windows.misc
Subject: leaking memory resources in 3.1
Message-ID: <29APR199309371113@bpavms.bpa.arizona.edu>
Date: 29 Apr 93 16:37:00 GMT
Organization: University of Arizona MIS Department
Lines: 46
NNTP-Posting-Host: bpavms.bpa.arizona.edu
News-Software: VAX/VMS VNEWS 1.41

This may be an FAQ (if so, please direct me to the known answer) but I am getting frustrated and looking for help.

I am running Win 3.1 with NDW 2.2 on a 486sx with 8 meg of memory and a 6 meg perm swap file and am getting exceedingly frustrated that my applications are not giving back system resources when I close them.

When I load windows I start with about 83% resources available but many of the application programs I run regularly (WPWIN 5.2, VB 2.0, WINQVT 2.8, Lotus Organizer, etc.) seem to not return all their resources when I exit them. After a day or two of work I often find myself down under 50% of available resources even when I have no applications other than my shell running.

I am aware this is a known problem; what I am looking for are some suggestions of what I might do to mitigate it.

1. What software is the culprit? Win 3.1, NDW, my applications? Are some modes of Win 3.1 (standard, real, enhanced) better than others at plugging this leak?

2. Are their system.ini switches i can set to help plug this leak?

3. Do people know of patches or third party software that help with this? Seems like increasing or better managing system resources is a great market for a third party memory company like QEMM.

Thanks for your help. As this is a common problem and I have seen only a little discussion of it on the net there are probably others who would like to read answers so please publish here rather than sending me email.

Gambar 1

12. Uji Coba Metode ID3

12.1. Deskripsi Data

Jumlah partisi: 10
Jumlah atribut: 2.000
Jumlah *training* data: 1.300 / partisi
Jumlah *test* data: 700 / partisi

12.2. Cara Melakukan Percobaan

Program yang digunakan untuk melakukan percobaan adalah WEKA. Eksperimen dilakukan sebanyak 10 kali sesuai dengan jumlah partisi yang dilakukan terhadap 20_newsgroups. Setiap percobaan dilakukan dengan memilih *training* data sebanyak 1.300 dan *test* data sebanyak 700 secara random dari 20 kategori.

12.3. Proses Klasifikasi

Proses pembentukan model untuk klasifikasi terhadap training data pada algoritma ID3 dilakukan dengan membuat model berupa *decision tree*. Program WEKA

yang digunakan untuk melakukan percobaan ini membangkitkan *decision tree* dalam waktu kurang dari 10 detik untuk setiap eksperimen yang dilakukan. Performansi yang diukur dalam percobaan ini adalah tingkat akurasi algoritma dalam melakukan klasifikasi teks secara benar ke dalam 20 kategori *newsgroups*. Berikut ini adalah nilai persentase akurasi pada 10 partisi *test* data yang digunakan:

Partisi ke-n	Akurasi (%)
1	77.5758
2	80.303
3	81.8182
4	76.3636
5	79.697
6	78.7879
7	80.6061
8	80.1515
9	78.1818
10	81.3636
AVG =	79.48485
STD =	1.730785842

13. Uji Coba Metode *Naive Bayes*

13.1. Deskripsi Data

Jumlah partisi: 10
Jumlah atribut: 2.000
Jumlah *training* data: 1.300 / partisi
Jumlah *test* data: 700 / partisi

13.2. Cara Melakukan Percobaan

Program yang digunakan untuk melakukan eksperimen adalah WEKA. Percobaan dilakukan sebanyak 10 kali sesuai dengan jumlah partisi yang dilakukan terhadap 20_newsgroups. Setiap percobaan dilakukan dengan memilih *training* data sebanyak 1.300 dan *test* data sebanyak 700 secara random dari 20 kategori.

13.3. Proses Klasifikasi

Program WEKA yang digunakan untuk melakukan percobaan ini dapat membuat model dari *training* data dalam waktu kurang dari 2 detik untuk setiap percobaan yang dilakukan. Performansi yang diukur dalam percobaan ini adalah tingkat akurasi algoritma dalam melakukan klasifikasi teks secara benar ke dalam 20 kategori *newsgroups*. Berikut ini adalah nilai persentase akurasi pada 10 partisi *test* data yang digunakan:

Partisi ke-n	Akurasi (%)
1	76.6667
2	75.7576
3	73.4848
4	73.1818
5	76.9697
6	72.1212
7	72.1212
8	71.5152
9	78.0303
10	75.7576
AVG =	74.56061
STD =	2.342318345

14. Uji Coba Metode *Artificial Neural Network*

14.1. Deskripsi Data

Jumlah partisi: 10
Jumlah atribut: 100
Jumlah *training* data: 1.300 / partisi
Jumlah *test* data: 700 / partisi
Jumlah *hidden layer*: 30
Jumlah iterasi: 500

14.2. Cara Melakukan Eksperimen

Program yang digunakan untuk melakukan percobaan adalah WEKA. Percobaan dilakukan sebanyak 10 kali sesuai dengan jumlah partisi yang dilakukan terhadap 20_newsgroups. Setiap percobaan dilakukan dengan memilih *training* data sebanyak 1.300 dan *test* data sebanyak 700 secara random dari 20 kategori.

14.3. Proses Klasifikasi

Program WEKA yang digunakan untuk melakukan percobaan ini dapat membuat model *neural network* dari *training* data dalam waktu kurang lebih 7 menit untuk setiap percobaan yang dilakukan. Performansi yang diukur dalam percobaan ini adalah tingkat akurasi algoritma dalam melakukan klasifikasi teks secara benar ke dalam 20 kategori *newsgroups*. Berikut ini adalah nilai persentase akurasi pada 10 partisi *test* data yang digunakan:

Partisi ke-n	Akurasi (%)
1	35.1515
2	31.2121
3	32.8788
4	27.7273
5	34.697
6	30.9091
7	28.0303
8	30.303
9	30.9091
10	30
AVG =	31.18182
STD =	2.476087191

15. Uji Coba Metode *K-Nearest Neighbor*

15.1. Deskripsi Data

Jumlah partisi: 10
Jumlah atribut: 2.000
Jumlah *training* data: 1.300 / partisi
Jumlah *test* data: 700 / partisi
Nilai k: 1

15.2. Cara Melakukan Percobaan

Program yang digunakan untuk melakukan percobaan adalah WEKA. Percobaan dilakukan sebanyak 10 kali sesuai dengan jumlah partisi yang dilakukan terhadap 20_newsgroups. Setiap eksperimen dilakukan dengan memilih *training* data sebanyak 1.300 dan *test* data sebanyak 700 secara random dari 20 kategori.

15.3. Proses Klasifikasi

Proses pembangunan model untuk klasifikasi terhadap training data pada algoritma *k-nearest neighbor* dilakukan dengan nilai k sama dengan satu. Program WEKA yang digunakan untuk melakukan percobaan ini dapat membuat model dari *training* data dalam waktu kurang dari 1 detik untuk setiap percobaan yang dilakukan.

Performansi yang diukur dalam percobaan ini adalah tingkat akurasi algoritma dalam melakukan klasifikasi teks secara benar ke dalam 20 kategori *newsgroups*. Berikut ini adalah nilai persentase akurasi pada 10 partisi *test* data yang digunakan:

Partisi ke-n	Akurasi (%)
1	43.4848
2	41.8182
3	40
4	42.7273
5	43.1818
6	40.4545
7	41.3636
8	37.5758
9	38.3333
10	38.3333
AVG =	40.72726
STD =	2.1410122695

16. Hasil Percobaan

16.1. Data *summary* percobaan

Hasil eksperimen dengan 4 jenis algoritma klasifikasi teks adalah sebagai berikut:

No	Algoritma Klasifikasi	Nilai Rata-Rata	Waktu Rata-
1	ID3	79.48485%	0,5 menit
2	K-Nearest Neighbor	40.72726%	2 menit
3	Naive Bayes	74.56061%	0,5 menit
4	Artificial Neural	31.18182%	9 menit

16.2. Analisis Hasil Percobaan

Algoritma ID3 cukup sesuai untuk melakukan klasifikasi dokumen karena model yang dihasilkan dari proses learning terhadap training data berupa pohon keputusan (*decision tree*). Dengan pemodelan *decision tree*, persoalan klasifikasi yang cukup sederhana dapat dilakukan dengan akurat karena dalam mengklasifikasikan suatu objek, tentunya terdapat aturan-aturan yang menjadi penentu termasuk kategori manakah objek tersebut.

Dengan algoritma ID3, pohon keputusan dibangun berdasarkan aturan-aturan tersebut, sehingga proses *learning* akan memiliki akurasi yang tinggi karena klasifikasinya dilakukan dengan melakukan inferensi terhadap aturan-aturan tersebut. Hal ini terbukti dengan nilai rata-rata akurasi yang mendekati 80% untuk setiap eksperimen.

Algoritma naive bayes cukup sesuai untuk melakukan klasifikasi dokumen karena pemodelan algoritma ini menggunakan pendekatan statistik. Kumpulan-kumpulan dokumen dalam kategori yang sama umumnya memiliki penggunaan kata-kata tertentu yang relatif sering muncul. Oleh karena itu, pemodelan statistik berdasarkan frekuensi kemunculan kata-kata dalam suatu dokumen akan memberikan akurasi yang tinggi pada proses klasifikasi dengan menggunakan algoritma naive bayes.

Algoritma *k-nearest neighbor* (KNN) kurang cocok untuk melakukan klasifikasi dokumen karena KNN tidak menghasilkan pemodelan dari proses *learning* terhadap training data namun hanya berdasarkan perhitungan jarak terpendek dari *query instance* ke *training sample*. Proses pembelajaran berdasarkan

perhitungan jarak menghasilkan representasi yang tidak jelas mengenai jenis jarak dan atribut mana yang harus digunakan untuk mendapatkan hasil yang terbaik. Selain itu biaya komputasi cukup tinggi karena diperlukan perhitungan jarak dari tiap *query instance* pada keseluruhan *training sample*. Dari eksperimen yang dilakukan, semakin tinggi nilai k maka diperoleh hasil pengklasifikasian yang lebih rendah dan waktu pemrosesan yang lebih lama.

Artificial neural network (ANN) kurang cocok untuk menyelesaikan persoalan yang cukup sederhana seperti klasifikasi dokumen karena hasil yang didapat tidak sebanding dengan lamanya waktu proses yang diperlukan. Berdasarkan eksperimen, performansi ANN akan meningkat jika jumlah atribut dan jumlah *hidden layer* ditambah, tetapi hal ini akan mengakibatkan waktu proses yang jauh lebih lama.

17. Kesimpulan

Kesimpulan yang dihasilkan dari kajian metode klasifikasi dokumen teks adalah:

1. Terdapat berbagai macam metode yang dapat dipakai untuk permasalahan klasifikasi dokumen teks.

2. *Preprocessing* pada proses uji coba merupakan proses yang sangat penting dan meliputi hampir 80% proses dalam melakukan klasifikasi dokumen teks. Dengan *preprocessing* yang benar, maka tingkat akurasi klasifikasi dokumen teks akan meningkat pula.

3. Metode pembentukan pohon ID3 dan Naive Bayes merupakan metode yang cukup sesuai untuk melakukan klasifikasi dokumen teks. Hal ini terlihat dari nilai rata-rata tingkat akurasi yang didapatkan dari hasil eksperimen yang mendekati 80%.

4. Algoritma ANN sebenarnya dirancang untuk persoalan rumit dimana pada persoalan tersebut biasanya tidak terdapat pola-pola yang dapat dikenali dan data yang digunakan untuk proses *training* adalah data yang memiliki tingkat *noise* yang tinggi sehingga ANN kurang memberikan hasil yang memuaskan untuk persoalan klasifikasi dokumen ini karena selain persoalannya memang relatif sederhana, data yang digunakan untuk proses *training* juga adalah data yang lengkap.

5. Nilai atribut untuk algoritma metode pembentukan pohon ID3, *k-nearest neighbor*, naive bayes, dan *artificial neural network* merupakan nilai biner yang menandakan apakah suatu kata muncul atau tidak di dalam dokumen. Dengan pemodelan dokumen seperti ini maka tingkat akurasi dan proses klasifikasi dapat berjalan dengan optimal.

6. Berdasarkan metode perbandingan statistik *two-tail t test*, metode ID3 dan naïve bayes dapat dikatakan *comparable* karena algoritma kedua metode ini memiliki nilai rata-rata akurasi yang hampir sama, yaitu berada di antara rentang nilai 74-80%.

7. Penggunaan metode Pohon ID3 merupakan metode yang tepat karena memiliki performa tertinggi dalam percobaan makalah ini.

DAFTAR PUSTAKA

[1] Mitchell, Tom M. (1997). Machine Learning. McGraw-Hill.

[2] Collins, Michael & Barzilay, Regina (2005). Lectures Presentation: Natural Language Processing, Background and Overview. Open-Course EECS/CSAIL, MIT.

[3] Munir, Rinaldi. (2003). Bahan Kuliah IF2153 Matematika diskrit. Departemen Teknik Informatika, Sekolah Teknik Elektro Informatika, Institut Teknologi Bandung.

[4] Victor, Ramadan, Riza, Widyasari, Wulan, Tanoto, Andri, Sajuthi, Satria P. (2006). Eksperimen Klasifikasi Teks. Departemen Teknik Informatika, Sekolah Teknik Elektro Informatika, Institut Teknologi Bandung.

[5] Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, P. J. (1984). *Classification and Regression tree*. Belmont, CA: Wadsworth International Group.