Implementasi *Complement Naive Bayes* untuk *Automated Essay Scoring System*

Mustaqiim Agam Santoso, Muhammad Azmi Fansuri, Muhamad Musta'in

Abstrak

E-learning menjadi platform yang telah banyak digunakan dalam pendidikan di era berkembang pesatnya teknologi. Pembelajaran melalui *e-learning* selain diharap dapat mempermudah fleksibilitas siswa dalam belajar, juga dapat mempermudah pekerjaan guru dalam melakukan evaluasi terhadap siswa khususnya dalam penilaian jawaban esai. *Automated Essay Scoring System* merupakan salah satu fitur yang dapat mempermudah guru untuk melakukan penilaian esai siswa secara otomatis, dan memudahkan siswa dalam mengevaluasi perkembangan pembelajarannya secara langsung. Klasifikasi data menggunakan *Complement Naive Bayes* adalah metode pembelajaran mesin yang cocok diterapkan untuk *text mining* dari esai siswa. Hasil yang didapatkan pada penelitian ini yaitu penggunaan *CountVectorizer* dan *TfidfTransformer* yang berpotensi menghasilkan akurasi yang tinggi dalam proses automasi. Agar prediksi yang dilakukan lebih akurat diberikan parameter tuning melalui *GridSearchCV.*

Kata Kunci : Esai, Natural Language Processing, Complement Naive Bayes, Python

1 Pendahuluan

Pendidikan merupakan kebutuhan yang terbilang primer bagi manusia. Di era yang memasuki Revolusi Industri 4.0, pendidikan baiknya dikolaborasikan dengan teknologi dengan harapan memberi efektivitas dan efisiensi pengajaran melalui media yang lebih menyenangkan bagi siswa. Salah satu media yang populer digunakan adalah *e-learning*.

Melalui *e-learning,* siswa dapat mengakses semua materi yang disediakan tanpa perlu memikirkan tempat dan waktu. Guru juga bisa lebih fleksibel dalam melakukan penilaian tugas siswa. Untuk tugas yang berbentuk pilihan ganda, melalui sistem pencocokan lewat komputer dapat dilakukan dengan mudah. Masalah terjadi apabila guru memberikan tugas esai kepada siswa. Untuk itu, banyak peneliti bidang Natural Language Processing (NLP) sekarang mencoba menerapkan pembelajaran mesin dalam melakukan penilaian tugas esai secara otomatis melalui *automated essay scoring system.* Dengan adanya fitur ini di dalam platform *e-learning,* diharapkan siswa dapat langsung mengetahui hasil evaluasi pekerjaannya. Selain itu, dari sisi guru juga dapat membantu dalam manajemen waktu dan kegiatan menjadi lebih efisien.

Penelitian ini bertujuan untuk melihat akurasi apabila diterapkan *automated essay scoring* dengan membandingkan hasilnya terhadap hasil penilaian guru. Jenis pembelajaran mesin yang digunakan dalam penelitian ini klasifikasi dari *text mining*. Metode yang digunakan adalah *CountVectorizer* dan *TfidfTransformer* yang bertujuan untuk mengubah teks menjadi vektor yang independen dan tidak saling berelasi antar entri data, sehingga masing-masing entri vektor tersebut dapat lebih mewakili label yang sudah disediakan dari penilaian guru. Teknik klasifikasi yang digunakan adalah *Complement Naive Bayes.*

Berdasarkan uraian tersebut, maka diperlukan fitur *automated essay scoring system* dalam *e-learning* untuk mengefisienkan waktu dan tenaga guru serta hasil evaluasi siswa dalam penilaian jawaban esainya.

2 Kajian Literatur

Metode yang dilakukan untuk melakukan prediksi pelabelan dalam eksperimen ini adalah klasifikasi data. Klasifikasi adalah salah satu teknik untuk melabeli dataset dengan beberapa label yang masing-masing memiliki makna tersendiri.

Dalam *text mining*, tahapan dimulai dengan *text pre-processing* seperti menghilangkan delimiter (Weiss et al, 2005). Selanjutnya, melakukan seleksi fitur untuk menghapus kata yang tidak penting atau tidak menggambarkan isi dokumen (Feldman & Sanger,2007., Berry & Kogan, 2010.). Tahap ini juga menghilangkan *stopword* yaitu kosakata yang bukan merupakan ciri unik dari dokumen (Dragut et al, 2009.).

Salah satu teknik klasifikasi yang familiar digunakan dalam *text mining* adalah Naive Bayes. Dalam eksperimen ini, penulis menggunakan algoritma *Complement Naive Bayes* yang merupakan teknik modifikasi dari *Naive Bayes.*

Tahapan dalam algoritma *Complement Naive Bayes* :

a. Menghitung probabilitas bersyarat (*likelihood)* :
$$P(x|C^*) = P(x_1, x_2, \dots, x_n|C^*)$$
dengan C^* adalah kelas komplemen dari C, dan x adalah vektor dari nilai atribut n.
Adapun $P(x_i|C^*)$ adalah proporsi dokumen dari kelas-kelas yang mengandung nilai atribut x_i di luar kelas C.

b. Menghitung probabilitas *prior* untuk tiap kelas:
$$P(C) = \frac{N_j}{N}$$
dengan N_j adalah jumlah kelas dari C dan N adalah jumlah total seluruh kelas.

c. Menghitung probabilitas *posterior* dengan rumus:
$$P(C|x) = \frac{P(C)}{P(x|C)P(x)}$$
atau dapat ditulis dalam bentuk kata lebih umum menjadi :
$$Posterior = \frac{prior}{likelihood \times evidence}$$

Pengklasifikasi *Complement Naive Bayes* yang merupakan bagian dari model *Naive Bates* ini merupakan teknik sederhana yang efisien dan dominan memberikan performa baik. Kekurangan teknik ini terletak pada sensitivitas yang tinggi terhadap fitur yang terlalu banyak yang menyebabkan akurasi rendah.

3 Metode Penelitian

Metode penelitian yang penulis lakukan dalam merancang *automatic essay scoring* ini adalah melalui eksperimen terhadap data dari kasus 2 esai dengan topik berbeda yang telah disediakan.

Esai pertama yang diberikan dalam dataset A memaparkan mengenai pemanasan global, sedangkan esai kedua yang diberikan dalam dataset B lebih memaparkan tentang studi kasus penarikan kesimpulan. Dari kedua dataset yang diberikan, terdapat dua klasifikasi penilaian yang bernilai 1 apabila jawaban yang diberikan siswa relevan dengan permasalahan yang diangkat, dan 0 jika terjadi sebaliknya. Tahapan awal sebelum memulai untuk klasifikasi data adalah melakukan pengolahan awal data.

3.1 Pengolahan Awal Data (*Text Pre-processing)*

Dalam tahap preprocessing, penulis membagi pengolahan data melalui 4 proses, yaitu :

1. *Tokenization*
   Membuat daftar semua kata yang muncul dalam setiap baris data dengan menghilangkan tanda baca ataupun simbol menggunakan fungsi *replace()* dan fungsi *split()* yang menggunakan spasi untuk memotong kalimat menjadi per kata, serta menghilangkan *blank element* yang muncul dalam list kata saat proses tokenisasi. Dalam tahapan ini juga dilakukan *case folding*, yaitu pengubahan semua karakter huruf menjadi huruf kecil (Garcia, 2005).
2. *Penentuan Stopwords*
   Menentukan kata-kata yang tidak relevan dengan topik yang diangkat dalam masing-masing dataset. Kata-kata yang dimaksud, contohnya adalah : “yang”, “akan”, serta “dan”.
   Stopwords yang digunakan penulis dalam eksperimen ini hanya menggunakan kata yang paling sering muncul yang termuat di kedua grup kalimat berdasarkan label klasifikasi.
   Adapun untuk keputusan dihapus atau tidaknya *stopwords* yang ada di dalam kalimat dilakukan pada tahapan berikutnya.

Setelah dilakukan *pre-processing* awal dari dataset, selanjutnya dilakukan klasifikasi menggunakan algoritma *Complement Naive*

Bayes. *Pre-processing* ini belum diterapkan ke dataset dengan tujuan untuk *parameter tuning* yang dilakukan pada tahap berikutnya.

3.2 Proses Algoritma *Complement Naive Bayes*

Algoritma Complement Naive Bayes (CNB) diadaptasi dari algoritma Multinomial Naive Bayes yang digunakan terhadap data-data yang tidak berimbang (*imbalanced data sets)*. Algoritma Multinomial Naive Bayes sendiri sebetulnya adalah model klasifikasi yang biasa digunakan dalam klasifikasi teks.

Dalam dataset yang diberikan, penulis menghitung banyaknya kalimat dengan label 0 atau 1 dari masing-masing dataset dan menemukan adanya ketidakseimbangan jumlah kalimat yang muncul. Misalnya, untuk dataset A yang terdiri dari 268 data terdapat 77 yang berlabel 0 dan 191 yang berlabel 1. Sedangkan untuk dataset B terdiri dari 305 data dengan 137 data berlabel 0 dan 168 data berlabel 1. Dengan adanya kasus tersebut, penulis merasa bahwa model CNB cocok dalam dataset ini. Sebelum memutuskan menggunakan model tersebut, penulis juga melakukan uji coba dalam model lain seperti *Multinomial Naive Bayes* tetapi menemukan bahwa akurasi tertinggi masih dicapai oleh CNB.

3.3 Penggunaan GridSearchCV untuk *Parameter Tuning*

Tahapan selanjutnya yang dilakukan penulis adalah membuat *pipeline* dari *GridSearchCV* yang mengkombinasikan *pre-processing* dengan model *classifier* yang telah dipilih, yaitu *Complement Naive Bayes*. Cara ini dilakukan dengan maksud agar penulis dapat mencoba beberapa tahapan *pre-processing* dan penentuan parameternya yang memberikan *f1-score* yang tinggi tanpa harus mencoba satu demi satu fase pengklasifikasian.

Pre-processing yang dilakukan di dalam *pipeline* tersebut menggunakan komponen *CountVectorizer.* Komponen ini digunakan untuk melakukan tokenisasi terhadap teks, yaitu mengubah fitur teks menjadi representasi vektor. Parameter yang digunakan dalam *pipeline* untuk komponen ini adalah *stop_words* untuk menghapus *stopwords* yang telah ditentukan dari tahapan sebelumnya, serta *ngram_range* yang digunakan untuk menggabungkan kata untuk melihat adanya frase yang menunjukkan sentimen atau tendensi kuat ke label tertentu. Untuk parameter *stopwords* diuji-coba apakah *f1-score* lebih bagus dengan menghilangkan *stopwords* dari teks atau tidak, sedangkan untuk parameter *ngram_range* diuji-coba untuk beberapa tupel berikut : (1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (2, 5), (3, 5), (4, 5), (5, 5).

Pre-processing yang dilakukan berikutnya di dalam *pipeline* tersebut menggunakan komponen *TfidfTransformer.* Komponen ini digunakan untuk melakukan transformasi vektor yang dibentuk dari *CountVectorizer* menjadi *normalized term-frequency* atau *term-frequency times inverse document-frequency*. *Term-frequency* sendiri merepresentasikan frekuensi kemunculan sebuah term dalam dokumen. Semakin sering suatu *term* muncul, maka semakin besar pembobotannya. Sedangkan *Inverse Document Frequency (IDF)* merupakan penghitungan bagaimana *term* didistribusikan secara luas di koleksi dokumen tersebut. IDF menunjukkan hubungan ketersediaan sebuah *term* dalam seluruh dokumen. Semakin sedikit jumlah dokumen yang mengandung *term* tersebut, nilai IDF semakin besar.

Parameter yang digunakan dalam *pipeline* untuk komponen *TfidfTransformer* adalah *use_idf* untuk memasukkan perhitungan IDF dalam analisis ataukah tidak. Selanjutnya, yaitu *norm* yang merupakan norma unit vektor yang digunakan dengan mengukur apakah lebih bagus menggunakan norma vektor di R1 atau R2. Penerapan *TfidfTransformer* di dalam *pipeline* merupakan tahap lanjut dari vektorisasi teks, yang mana apabila hanya dilakukan *CountVectorizer* kurang tepat karena memungkinkan ada vektor yang masih berelasi satu sama lain. *TfidfTransformer* menghilangkan relasi antar kata.

Setelah membentuk *pipeline* dan mendefinisikan parameter percobaan untuk diterapkan dalam masing-masing komponen di dalam *pipeline*, fungsi *GridSearchCV* dibentuk dengan memperhatikan nilai-nilai parameter tertentu dari fungsi tersebut. Misalnya, parameter *n_jobs* untuk menentukan jumlah pemrosesan data yang dikerjakan secara paralel. Parameter ini menggunakan nilai -1 menggunakan semua prosesor. Parameter lainnya adalah *verbose* untuk mengatur *logging information* dan menggunakan nilai 1. Semakin tinggi nilai yang diberikan di parameter *verbose*, akan semakin rinci log informasi yang disajikan.

3.4 Evaluasi dan Validasi Hasil

Validasi hasil dilakukan menggunakan GridSearchCV dengan 5 *fold cross validation* yang akan membagi dataset ke dalam 5 model data-splitting yang 1/5 datanya digunakan untuk testing. Sedangkan untuk akurasi diukur melalui *confusion matrix* dan *f1-score*. *Confusion matrix* digunakan untuk mengukur performansi dari metode klasifikasi yang akan lebih jelas menunjukkan angka perbandingan hasil label menggunakan metode klasifikasi dengan label yang sebenarnya. Berikut adalah bentuk dari *confusion matrix* beserta rumus perhitungan menurut Gorunescu (Gorunescu, 2011):

Dari *confusion matrix* tersebut, dapat dihitung metrik seperti *recall, precision,* dan *f1-score* dengan

Klasifikasi	**Hasil Prediksi**		
Hasil Observasi		**Ya**	**Tidak**
	Ya	(*True Positive – TP)*	(*False Negative – FN)*
	Tidak	(*False Positive – FP)*	(*True Negative – TN)*

Table 1: Confusion Matrix.

$$Recall = \frac{TP}{TP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$F1\ score = \frac{2TP}{2TP + FP + FN}$$

4 Hasil dan Pembahasan

Pengujian dilakukan dengan menghitung nilai *precission, recall,* dan *f1-score* dari data training yang dilakukan secara terpisah untuk dataset A dan B. Sebelum melakukan pengujian, *pre-processing* teks yang dilakukan menghasilkan data baru setelah tokenisasi dan *stopwords removal* dengan contoh dari data A pada indeks ke-247 sebagai berikut.

Data	**Tokenisasi**
'akan mendapatkan negara yang bagus dan aman.'	'akan mendapatkan negara yang bagus dan aman'

Table 2: Tokenisasi

Pengelompokan kata yang termasuk *stopwords* dilakukan dengan menghimpun 10 kata-kata yang paling sering muncul di data berlabel 0 dan berlabel 1 dari dataset A sebagai berikut.

Common words label 0		**Common words label 1**	
yang	38	mereka	178
akan	29	yang	142
mereka	23	akan	113
bencana	16	baru	106
karena	15	dan	84
iklim	15	tempat	82
pengungsi	15	dengan	77
tempat	13	beradaptasi	63
dan	13	lingkungan	59
komunitas	12	di	49

Table 3: Common words dataset A

Selanjutnya dari kumpulan kata tersebut dipilih kata yang muncul di kedua label, yaitu kata : ‘yang’, ‘akan’, ‘mereka’, ‘tempat’, serta ‘dan’. Sedangkan, untuk kata-kata yang paling sering muncul untuk dataset B adalah sebagai berikut.

Common words label 0		**Common words label 1**	
karena	85	mereka	124
yang	70	yang	113
lebih	57	akan	102
pakaian	56	baru	78
untuk	44	dan	61

Table 4: Common words dataset B

Dari hasil tersebut, diperoleh *stopwords* untuk dataset B sebagai berikut : ‘karena’, ‘yang’, ‘lebih’, ‘pakaian’, dan ‘untuk’. Kata-kata tersebut dikumpulkan menjadi satu list *stopwords* dan digunakan untuk *parameter tuning* dalam model *GridSearchCV.* Pengelompokan *common words* yang hanya muncul di kedua label menjadi *stopwords* dimaksudkan penulis untuk mengurangi peran dari kata-kata tersebut di masing-masing label. Dengan menghilangkan *common words* yang ada di kedua label tersebut, diharapkan antar label memiliki kumpulan kata yang benar-benar *distinct* atau independen satu sama lain sehingga kata yang tersisa lebih representatif terhadap labelnya. Dengan demikian, *F1 Score* yang dapat diperoleh dapat lebih meningkat. Di tahap berikutnya, apabila diterapkan *stopwords removing,* maka data yang baru dapat dimisalkan seperti berikut.

Data	Tokenisasi
'akan mendapatkan negara yang bagus dan aman.'	'akan mendapatkan negara yang bagus dan aman'

Table 5: Stopwords Removal

Selanjutnya, menggunakan *GridSearchCV* yang memuat pengklasifikasi CNB dan Pipeline yang telah didefinisikan, dilakukan beberapa kali evaluasi dengan maksud *parameter tuning* dengan *5 fold Cross Validation*.

Sebagai hasilnya, parameter optimal yang menghasilkan *F1 score* terbaik untuk data A adalah dengan *ngram_range* (1,4), menghapus *stopword,* menggunakan norma R1 dalam tf-idf dan tidak menggunakan idf. Sedangkan untuk data B parameter optimal yaitu dengan *n_gram range* (1,1) tanpa menghapus *stopword,* menggunakan norma R2 dalam tf-idf dan tidak menggunakan idf.

Setelah memperoleh parameter optimal, dilakukan prediksi pelabelan dan dilakukan evaluasi melalui *confusion matrix* untuk membandingkan hasil prediksi dengan hasil observasi. Diperoleh tabel *confusion matrix* untuk dataset A sebagai berikut.

Klasifikasi	Hasil Prediksi		
Hasil Observasi		**Ya**	**Tidak**
	Ya	63	14
	Tidak	6	185

Table 6: Confusion Matrix dataset A.

Sedangkan, untuk dataset B diperoleh sebagai berikut.

Klasifikasi	Hasil Prediksi		
Hasil Observasi		**Ya**	**Tidak**
	Ya	107	30
	Tidak	11	157

Table 7: Confusion Matrix dataset B.

Diberikan pula *performance metrics* dengan memperhatikan *precission, recall,* dan *F1 Score* dari masing-masing label di setiap dataset sebagai berikut untuk menunjukkan seberapa baik model terbentuk.

Dataset	Precision	Recall	F1-score
Label 1 dataset A	93 %	97 %	95 %
Label 0 dataset A	91 %	82 %	86 %
Label 1 dataset B	84 %	93 %	88 %
Label 0 dataset B	91 %	78 %	84 %

Table 8: Performance metrics.

Secara keseluruhan, skor F1 yang diperoleh di dataset A adalah sekitar 95 % sementara skor F1 yang diperoleh di dataset B adalah sekitar 88 %. Prosentase F1 untuk dataset B lebih rendah dibanding A dikarenakan cukup banyak data yang memuat kalimat yang typo atau yang masih membutuhkan *cleansing* atau *pre-processing* yang lebih baik lagi. Misalnya, untuk kalimat

"TRB264,karena'pakaianawetlebihmahal.,0"

atau

"TRB200,karan menyumbang perbuatan baik dan menambah pahala bagi kita juga.,1"

Perbedaan penggunaan kata "karna", "karana", "karena" kemungkinan berpotensi membuat pelabelan menjadi kurang terklasifikasi dengan benar karena ketiga kata tersebut dianggap sebagai kata berbeda yang berarti berpotensi menjadi nilai pembobotan yang berbeda dalam pemodelan.

Implikasi yang diperoleh dari eksperimen ini adalah dengan menunjukkan bahwa *Complement Naive Bayes* merupakan teknik klasifikasi yang cocok untuk klasifikasi data dengan banyaknya data per label tidak berimbang. Penerapan metode ini diharapkan dapat mempermudah para guru atau instruktur dalam menilai tugas esai siswa.

5 Kesimpulan

Dari pengolahan data yang telah dilakukan, terbukti bahwa *Complement Naive Bayes* dapat menjadi teknik yang potensial untuk analisis sentimen dari teks atau klasifikasi teks. Penerapan *tuning* untuk parameter dan dalam *pre-processing* melalui *GridSearchCV* juga berpotensi mempermudah simulasi dan evaluasi untuk menemukan parameter yang optimal.

Melalui analisis yang dilakukan pada dataset A, dapat disimpulkan bahwa dengan menghapus *stopwords* berpotensi untuk meningkatkan *F1 Score*. Sedangkan untuk kasus-kasus tertentu, menghapus *stopwords* justru menurunkan *F1 Score* sebagaimana eksperimen yang penulis lakukan dalam dataset B. Untuk eksperimen ke depan, dibutuhkan teknik khusus untuk menangani *typo* atau tidak adanya spasi dalam suatu kalimat secara otomatis. Hal ini diharapkan dapat meningkatkan *F1 Score.*

Meski demikian, model yang terbentuk ini dapat diterapkan untuk review atau penilaian esai dan melihat apakah tulisan siswa relevan dengan topik atau tidak. Hal ini tentu akan membantu guru atau pengajar dalam manajemen waktu dan tugasnya. Pengumpulan esai dalam bentuk digital

yang dimaksudkan sebagai *data collection* untuk model ini juga dapat mengantisipasi adanya tulisan yang tidak terbaca apabila tugas ditulis tangan.

Hasil dan sumber lengkap untuk metode eksperimen serta evaluasi yang dilakukan dimuat oleh penulis pada laman https://github.com-/agammsantos/Ukara-1-Final untuk kebutuhan lebih lanjut.

References

Abtohi, S. 2017. Simulasi Pembobotan Kata dengan TF-IDF. Available: http://www.analisis-data.com/2017/07/simulasi-pembobotan-kata dengan-tf-idf.html

Berry, M.W. & Kogan, J. 2010. *Text Mining Aplication and theory*. WILEY : United Kingdom.

Brownlee, J. 2016. What is a Confusion Matrix in Machine Learning. Available: https://machine-learningmastery.com/confusion-matrix-machine-learning/

Dragut, E., Fang, F., Sistla, P., Yu, S. & Meng, W. 2009. Stop Word and Related Problems in Web Interface Integration. http://www.vldb.org/pvldb/2/vldb09-384.pdf.

Feldman, R & Sanger, J. 2007. *The Text Mining Handbook : Advanced Approaches in Analyzing Unstructured Data*. Cambridge University Press : New York.

Garcia, E. 2005. Document Indexing Tutorial for Information Retrieval Students and Search Engine Marketers. Available: http://www.miislita.com/ information-retrieval-tutorial/indexing.html

Pedregosa et al. 2011. *Scikit-learn: Machine Learning in Python*. JLMR 12, pp.2825-2830

Vickery, R. 2019. A Simple Guide to Scikit-learn Pipelines. Available: https://medium.com-/vickdata/a-simple-guide-to-scikit-learn-pipelines-4ac0d974bdcf

Weiss, S.M., Indurkhya, N., Zhang, T., Damerau, F.J. 2005. *Text Mining : Predictive Methods for Analyzing Unstructured Information.* Springer : New York.