IndoMS Journal on Industrial and Applied Mathematics
Volume. **2**, Issue. 1 (2015), pp. 15-25

UJI KINERJA *LEARNING TO RANK* DENGAN METODE *SUPPORT VECTOR REGRESSION*

ABDUL AZIS ABDILLAH*, HENDRI MURFI†, DAN YUDI SATRIA‡

Ringkasan. Aplikasi *ranking* telah digunakan secara luas dalam kehidupan sehari-hari. Salah satu aplikasi *ranking* terdapat pada *search engine*. Walaupun hingga saat ini telah banyak dikembangkan metode *ranking*, akan tetapi hasil yang diperoleh masih belum optimal, dengan kata lain jika seorang pengguna memasukkan suatu *query* pada sebuah *search engine*, dokumen-dokumen yang ditampilkan belum tentu merupakan dokumen yang dibutuhkan oleh pengguna. *Learning to Rank* merupakan suatu metode yang menggunakan teknologi *machine learning* untuk menyelesaikan masalah *ranking*. Pada paper ini, peneliti mengusulkan uji kinerja *learning to rank* dengan metode *Support Vector Regression* (SVR). Uji coba menggunakan dataset LETOR MQ2008 dengan desain eksperimen *five-fold cross validation*. Hasil kinerja *learning to rank* yang diperoleh dengan menggunakan metode SVR dengan kernel RBF pada dataset LETOR MQ2008 dengan nilai parameter yang digunakan belum memperoleh hasil yang maksimal. Pada penelitian selanjutnya dapat dikembangkan menggunakan jenis kernel yang berbeda yaitu kernel linear dan polynomial.

Kata kunci. *Ranking*, *Learning to Rank*, SVR, LETOR MQ2008

1. Pendahuluan. Informasi yang telah tersimpan dalam suatu gudang informasi dengan jumlah yang sangat banyak tentu memerlukan suatu mekanisme agar informasi tersebut dapat dimunculkan kembali dan dapat dimanfaatkan oleh seseorang pengguna. seiring dengan berjalannya waktu, informasi semakin bertambah banyak, sehingga terjadilah banjir informasi.

Pada tahun 2005. *Yahoo!* mengumumkan bahwa *Search Engine Yahoo!* telah mengindeks lebih dari 19.2 milyar dokumen [11]. Informasi dengan jumlah yang sangat banyak tersebut, tentu memerlukan suatu mekanisme agar pengguna dapat melakukan pencarian informasi atau mendapatkan kembali informasi yang sesuai dengan kebutuhan secara cepat dan mudah. Tanpa hal tersebut, informasi yang melimpah tersebut akan tanpa guna.

Salah satu cara untuk mendapatkan kembali informasi yang sesuai dengan kebutuhan pengguna adalah dengan melakukan *Ranking* [6][10][11]. *Ranking* adalah merupakan bagian penting dari masalah pencarian informasi, seperti pengambilan dokumen, penyaringan informasi, penempatan iklan online, dan lain-lain [6][10][11]. Salah satu aplikasi *ranking* terdapat pada *search engine*, contohnya pada *Google* dan *Yahoo!* yang sudah sangat familiar dikalangan masyarakat.

Walaupun hingga saat ini telah banyak dikembangkan metode *ranking*, akan tetapi hasil yang diperoleh masih belum optimal [10][11], dengan kata lain jika seorang pengguna memasukkan suatu *query* pada sebuah *search engine*, dokumen-dokumen yang ditampilakan oleh *search engine* belum tentu merupakan dokumen yang dibutuhkan oleh pengguna. Sebagai contoh dari belum optimalnya metode *ranking* yang digunakan pada *search engine* diberikan pada gambar 1.1 [7]. Terlihat pada gambar 1.1 dari delapan dokumen yang ditampilkan oleh *search engine*, hanya dokumen ke satu, dokumen ke tiga dan dokumen ke tujuh yang dipilih oleh pengguna.

Salah satu solusi untuk menyelesaikan masalah *ranking* adalah dengan *machine learning* [10][11]. *Machine Learning* adalah metode atau model yang dapat belajar dari data historis sehingga menjadi cerdas. Cerdas dalam artian memiliki kemampuan generalisasi terhadap data baru yang belum dipelajari sebelumnya, misal untuk memprediksi, mengklasifikasi, *ranking*, dan lain-lain. Dengan menggunakan *machine learning* orang mencoba untuk mempelajari pola pengguna dalam menentukan relevansi suatu *web* terhadap suatu *query* yang diberikan. Topik ini dikenal dengan nama *Learning to Rank*. *Learning to rank* adalah suatu masalah *machine learning* yang bertujuan untuk membangun suatu model perangkingan dari data pembelajaran sehingga diperoleh urutan yang memiliki relevansi dan preferensi yang optimal [6][11].

Pada tahun 2010 *Yahoo!* mengadakan kompetisi untuk menyelesaikan masalah *learning to rank* dan mempublikasikan dataset LETOR yang menjadi standar baku penelitian pada topik *learning to rank* [2]. Metode yang telah digunakan antara lain Linear Regresi, Ranking SVM, Rank Boost, FRank, ListNet, AdaRank dan SVMMap [6][11]. Ide yang di ujicobakan oleh Cossock dan Zhang [3][6][11] yaitu dengan mengaitkan ke bentuk regresi, namun secara umum hasil yang diperoleh tidak terlalu baik [6][11]. Berdasarkan ide dari Cossock dan Zhang, pada paper ini peneliti mengusulkan menggunakan metode *Support Vector Regression* (SVR) untuk menyelesaikan persoalan *learning to rank*.

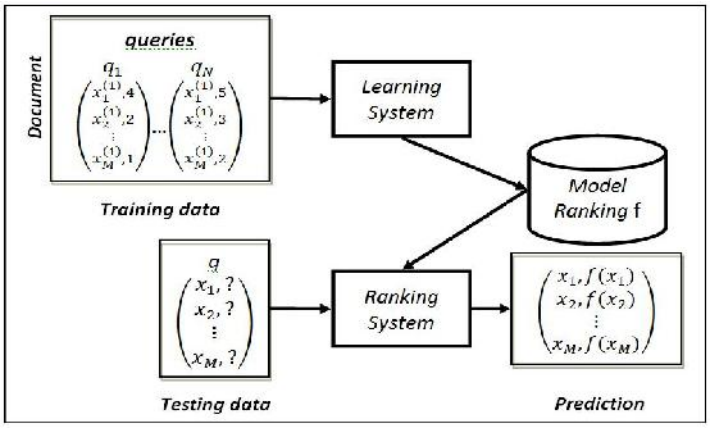
*Program Studi Pendidikan Matematika, STKIP Surya, Tangerang 15810 Banten, Indonesia (abdillah.azul@gmail.com).

†Departemen Matematika, Fakultas MIPA, Universitas Indonesia, Depok 16424, Indonesia (hendri@ui.ac.id).

‡Departemen Matematika, Fakultas MIPA, Universitas Indonesia, Depok 16424, Indonesia (ysatria@sci.ui.ac.id).

GAMBAR 1.1. *Hasil ranking yang ditampilkan oleh search engine dengan query "support vector machines" dan dokumen yang dipilih oleh pengguna*

2. Learning to Rank. Beberapa tahun terakhir, banyak sekali dikembangkan metode-metode *machine learning* yang digunakan untuk menyelesaikan masalah *ranking*. Menggunakan *machine learning*, orang mencoba untuk mempelajari pola pengguna dalam menentukan relevansi suatu web terhadap suatu *query* yang diberikan berdasarkan data historis yang dimiliki oleh *search engine*. Topik ini dikenal dengan nama *Learning to Rank*. Secara umum, *learning to rank* adalah suatu metode yang menggunakan teknologi *machine learning* yang bertujuan untuk secara otomatis membangun model *ranking* dari data pembelajaran sehingga diperoleh urutan yang memiliki relevansi dan preferensi yang optimal [10][11]. Secara umum tahapan proses *learning to rank* [6][9][11] dapat dilihat pada gambar 2.1.



GAMBAR 2.1. *Skema learning to rank*

Berdasarkan gambar 2.1 data *training* dan data *testing* pada *learning to rank* terdiri dari pasangan *query* $q = \{q_i\}_{i=1,2,...,N}$ dan data yang terdiri dari vektor *feature* $\mathbf{x} = \{x_j\}_{j=1,2,...M}$ beserta label relevansi. Vektor *feature* merupakan suatu kombinasi metode-metode *ranking* konvensional yang dijadikan sebagai elemen-elemen yang terdapat pada setiap vektor. Beberapa contoh metode *ranking* konvensional antara lain *Vector Space Model* (VSM), *Term Frequency* (TF), *Boolean Model* (BM25), *Language Model for Information Retrieval* (LMIR), *Pagerank*, dan lain-lain [6][11].

Secara umum tahapan *learning to rank* terdiri dari dua tahap yaitu *learning system* dan *ranking system* [6][11]. Pada tahapan *learning system* dilakukan pelatihan menggunakan suatu metode *ranking* terhadap data *training*, dengan tujuan untuk memperoleh suatu model *ranking* yang paling optimal, optimal dalam artian memperoleh akurasi *ranking* yang paling tinggi. Pada tahap selanjutnya, setelah mendapatkan model *ranking* yang paling optimal, model *rangking* tersebut diujikan pada data *testing* yang memiliki *query* baru, dengan tujuan untuk mendapatkan hasil urutan sebagai output dari pemilihan *query* baru tersebut.

TABEL 2.1
Konsep learning to rank untuk masalah regresi

	Metode Regresi	Ranking
Input	feature vector $\mathbf{x}$	feature vector $\mathbf{x} = \{x_i\}_{i=1}^{n}$
Output	bilangan real $y = f(\mathbf{x})$	urutan sort $(\{f(x_i)\}_{i=1}^{n})$
Model	Model Regresi $f(\mathbf{x})$	model ranking $f(\mathbf{x})$
Loss	regresi loss	ranking loss

Salah satu metode yang dapat dipakai untuk membangun model *rangking* pada *learning to rank* adalah dengan metode regresi [3][6][11]. Konsep *learning to rank* dengan menggunakan metode regresi [6][11] dapat di lihat pada Tabel 2.1.

3. Support Vector Regression. *Support Vector Regression* (SVR) adalah penerapan *Support Vector Machines* (SVM) untuk masalah regresi [1][4][8]. Dalam kasus klasifikasi output data berupa bilangan bulat atau diskrit, sedangkan pada masalah regresi, output data berupa bilangan real atau kontinu [1][4][8]. Misalkan $S = \{(\mathbf{x}_i, t_i)\}_{i=1,\ldots,m}$ merupakan suatu himpunan dengan m buah data dimana $\mathbf{x}_i \in \mathbb{R}^n$ dan $t_i \in \mathbb{R}$. Fungsi linier yang digunakan sebagai fungsi regresi adalah sebagai berikut [1][4][8]:

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b \tag{3.1}$$

dimana $\mathbf{x}$ adalah vektor input yang berupa data, $\mathbf{w}$ adalah vektor bobot, dan b adalah suatu bias. Tujuan pada kasus regresi adalah untuk menemukan suatu fungsi $f(\mathbf{x})$ yang memberikan nilai ε (error) paling minimum [1][4][8]. Karena fungsi keputusan $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$ tergantung pada parameter $\mathbf{w}$ dan b, maka yang harus dicari adalah nilai-nilai optimal dari $\mathbf{w}$ dan b yang dapat meminimumkan error, misalkan $\mathbf{w}^*$ dan b^*. SVR hanya dapat dipakai untuk menyelesaikan masalah yang sifatnya *linearly separable.* untuk menyelesaikan masalah nonlinier SVR, digunakan fungsi Kernel [1][4][8].

Pada kasus nonlinier SVR pertama-tama data di ruang input X dipetakan oleh fungsi transformasi Φ ke ruang Feature F, dimana dimensi dari F lebih besar dari dimensi X. Pada kasus data yang nonlinier, memetakan data dari ruang input ke ruang *Feature* dapat membuat data terpisahkan oleh sebuah *hyperplane* [8]. Dengan demikian model linier baru yang digunakan sebagai fungsi regresi memiliki bentuk umum sebagai berikut [1][4][8]:

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{\Phi}(\mathbf{x}) + b \tag{3.2}$$

dimana $\mathbf{x}$ adalah vektor input yang berupa data, $\mathbf{w}$ adalah vektor bobot, $\Phi(\mathbf{x})$ adalah fungsi transformasi $\mathbf{x}$ ke ruang *Feature* oleh Φ, dan b adalah suatu bias.

Pada kasus nonlinier, fungsi error ε-insensitive didefinisikan sebagai [1][4][8]:

$$E(\mathbf{w}, b) = C\sum_{i=1}^{m} E_\epsilon(\mathbf{w}^T\mathbf{\Phi}(\mathbf{x_i}) - t_n) + \frac{1}{2}\|\mathbf{w}\|^2 \tag{3.3}$$

Masalah optimisasi yang digunakan sebagai penentu parameter dan pada kasus nonlinier SVR dapat diformulasikan sebagai berikut [1][4][8]:

$$\min_{w,b,\xi,\widehat{\xi}} C\sum_{n=1}^{N}(\xi_n + \widehat{\xi}_n) + \frac{1}{2}\|\mathbf{w}\|^2 \tag{3.4}$$

dengan kendala :

$$t_i \leq f(\mathbf{x}_i) + \varepsilon + \xi_i \tag{3.5}$$

$$t_i \geq f(\mathbf{x}_i) - \varepsilon - \widehat{\xi}_i \tag{3.6}$$

$$\xi_i, \widehat{\xi}_i \geq 0 \tag{3.7}$$

dimana $f(\mathbf{x_i}) = \mathbf{w}^T \mathbf{\Phi}(\mathbf{x_i}) + b$ dan konstanta $C > 0$ menentukan tawar menawar (*trade off*) antara ketipisan fungsi f dan batas atas deviasi lebih dari ε yang masih bisa di toleransi. Fungsi Lagrange dari dari masalah optimisasi nonlinier SVR yaitu [1][4][8]:

$$L = C\sum_{i=1}^{m}\left(\xi_i + \hat{\xi}_i\right) + \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{m}\left(\mu_i\xi_i + \hat{\mu}_i\hat{\xi}_i\right) -$$

$$\sum_{i=1}^{m} a_i\left(\varepsilon + \xi_i + \mathbf{w}^T\Phi(\mathbf{x}_i) + b - t_i\right) -$$

$$\sum_{i=1}^{m} \hat{a}_i\left(\varepsilon + \hat{\xi}_i + \mathbf{w}^T\Phi(\mathbf{x}_i) + b - t_i\right) \tag{3.8}$$

dimana $a_i \geq 0$, $\hat{a}_i \geq 0$, $\mu_i \geq 0$, dan $\hat{\mu}_i \geq 0$ merupakan pengali Lagrange. Dengan menurunkan terhadap $\mathbf{w}$, b, ξ_i, dan $\hat{\xi}_i$ sama dengan nol, maka:

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{0} \rightarrow \mathbf{w} = \sum_{i=1}^{m}(a_i - \hat{a}_i)\Phi(\mathbf{x}_i) \tag{3.9}$$

$$\frac{\partial L}{\partial b} = 0 \rightarrow 0 = \sum_{i=1}^{m}(a_i - \hat{a}_i) \tag{3.10}$$

$$\frac{\partial L}{\partial \xi_i} = 0 \rightarrow C = a_i + \mu_i \tag{3.11}$$

$$\frac{\partial L}{\partial \hat{\xi}_i} = 0 \rightarrow C = \hat{a}_i + \hat{\mu}_i \tag{3.12}$$

Substitusikan persamaan (3.9), (3.10), (3.11), dan (3.12) ke persamaan (3.8), sehingga dihasilkan fungsi dual Lagrange dari masalah optimisasi nonlinier SVR yaitu [1][4][8]:

$$\widetilde{L}(\mathbf{a}, \widehat{\mathbf{a}}) = -\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}(a_i - \widehat{a}_i)(a_j - \widehat{a}_j)k(\mathbf{x}_i, \mathbf{x}_j) -$$

$$\varepsilon\sum_{i=1}^{m}(a_i + \widehat{a}_i) + \sum_{i=1}^{m}(a_i - \widehat{a}_i)t_i \tag{3.13}$$

Selanjutnya proses pembelajaran pada SVR dalam menemukan data yang menjadi *support vectors*, berkaitan dengan *dot product* dari data yang sudah di transformasikan ke ruang *Feature* yaitu $\Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j)$. Karena umumnya untuk mendapatkan fungsi transformasi Φ yang tepat sulit untuk diketahui, maka perhitungan *dot product* tersebut digantikan dengan fungsi Kernel $K(\mathbf{x}_i, \mathbf{x}_j)$. Proses penggantian dot product $\Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j)$ dengan fungsi kernel $K(\mathbf{x}_i, \mathbf{x}_j)$ disebut sebagai Kernel trick [1][4][8].

Dengan menggunakan fungsi Kernel maka *dot product* pada ruang *Feature* berdimensi tinggi atau bahkan berdimensi tak hingga dapat dihitung secara langsung dari ruang input tanpa harus secara eksplisit menghitung fungsi transformasi Φ. Dengan kata lain, tujuan digunakannya fungsi Kernel adalah untuk menggambarkan *dot product* di ruang *Feature* [1][4][8]. Fungsi Kernel yang digunakan pada percobaan ini adalah Fungsi Kernel Radial Basis Function yang didefinisikan sebagai [1][4][8]:

$$K(\mathbf{x}, \mathbf{y}) = exp\left(\frac{-\|\mathbf{x} - \mathbf{y}\|^2}{\sigma^2}\right)$$

dimana $\sigma > 0$.

Dengan menggunakan Kernel Trick maka bentuk dual nonlinier SVR di Ruang *Feature* dapat dinyatakan sebagai berikut [1][4][8]:

$$\max \widetilde{L}(\mathbf{a}, \widehat{\mathbf{a}}) = -\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}(a_i - \widehat{a}_i)(a_j - \widehat{a}_j)k(\mathbf{x}_i, \mathbf{x}_j) -$$

$$\varepsilon\sum_{i=1}^{m}(a_i + \widehat{a}_i) + \sum_{i=1}^{m}(a_i - \widehat{a}_i)t_i \tag{3.14}$$

dengan kendala :

$$0 \leq a_i, \hat{a}_i \leq C \tag{3.15}$$

$$\sum_{i=1}^{m}(a_i - \widehat{a}_i) = 0 \tag{3.16}$$

dengan menggunakan Kernel trick, maka fungsi keputusan nonlinier SVR di Ruang *Feature* dapat dinyatakan sebagai [1][4][8]:

$$f(\mathbf{x}) = \sum_{j \epsilon SV}^{m}(a_j^* - \widehat{a}_j^*)k(\mathbf{x}, \mathbf{x}_j) + b^* \tag{3.17}$$

dengan b^*:

$$b^* = t_i - \varepsilon - \sum_{j \epsilon SV}^{N}(a_j^* - \widehat{a}_j^*)k(\mathbf{x}_i, \mathbf{x}_j) \tag{3.18}$$

dimana i, j menunjukkan indeks dari data yang menjadi *support vectors* dan SV merupakan himpunan indeks dari *support vectors*. Selanjutnya, fungsi keputusan pada persamaan (3.17) digunakan sebagai model *ranking* yang digunakan pada kasus *learning to rank*.

4. Eksperimen. Percobaan *learning to rank* dengan metode *Support Vector Regression* (SVR) menggunakan *software* SVMlight, dilakukan pada dataset LETOR. Spesifikasi komputer yang digunakan pada percobaan ini adalah *Processor Intel Pentium(R) Dual-Core T4200 @ 2.0GHz 1.20 GHz*, RAM 3.5 GB dan OS Windows XP SP3.

4.1. Dataset. Pada percobaan ini dataset yang digunakan adalah dataset LETOR. LETOR merupakan koleksi data untuk *learning to rank* yang terbuka untuk umum (*open source*) dan diorganisasi oleh Microsoft. Lebih khusus lagi dataset yang digunakan adalah LETOR 4, MQ2008. Dataset MQ2008 merupakan kumpulan dokumen yang berasal dari websites dengan domain ".gov" [11].

Desain Eksperimen dari penelitian ini menggunakan *five-fold cross validation*. Dataset MQ2008 dibagi menjadi lima bagian dengan jumlah *query* sama untuk setiap bagian, setiap bagian dituliskan sebagai S1, S2, S3, S4, S5 dan kemudian disusun menjadi satu *fold*. Untuk setiap fold, tiga bagian digunakan untuk melatih model *ranking*, satu bagian digunakan untuk memvalidasi model *ranking* dan sisanya digunakan untuk mengevaluasi model *ranking*. Hasil rata-rata akurasi yang dicapai oleh lima *fold* tersebut digunakan sebagai akurasi *learning to rank*.

4.2. Ukuran Evaluasi Ranking. Kualitas relevansi hasil pencarian dengan perangkingan dokumen ditentukan oleh tiga nilai yaitu *Precision at position n*, *Mean Average Precision* dan *Normalized Discount Cumulative Gain* [6][9][11].

4.2.1. Precision at Position n ($P@n$). *Precision at position n* ($P@n$) merupakan ukuran evaluasi pada n dokumen teratas dengan suatu *query* q yang berasal dari hasil *ranking* menggunakan dua label relevansi yaitu *relevant* dan *irrelevant* [6][9][11], formula $P@n$ dituliskan sebagai berikut:

$$P@n = \frac{\#\, relevant\, documents\, in\, top\, n\, results}{n} \tag{4.1}$$

Karena terdapat tiga label relevansi pada dataset MQ2008 dan $P@n$ hanya dapat mengukur dua label relevansi, maka data yang memiliki label relevansi *highly relevant* akan dianggap sebagai *relevant* atau dapat dituliskan sebagai berikut:

$$rel(n) = \begin{cases} 1, & jika\, dokumen\, ke\, n^{th}\, relevant \\ 0, & jika\, lainnya \end{cases} \tag{4.2}$$

Sebagai contoh, jika mengambil 10 dokumen teratas hasil perangkingan dengan suatu *query* dengan hasil $2, 1, 1, 0, 2, 0, 1, 1, 00$, maka $P@1$ sampai $P@10$ bernilai

$$\{1, \frac{2}{2}, \frac{3}{3}, \frac{3}{4}, \frac{4}{5}, \frac{4}{6}, \frac{5}{7}, \frac{6}{8}, \frac{6}{9}, \frac{6}{10}\}$$

dan untuk mendapatkan nilai $P@n$ keseluruhan, maka harus dicari nilai rata-rata $P@n$ untuk semua *query*.

4.2.2. Mean Average Precision (MAP). Untuk setiap *query*, *Average Precision* (AP) merupakan nilai rata-rata $P@n$ yang relevan untuk setiap *query* [6][9][11], formula AP dituliskan sebagai berikut :

$$AP = \frac{\sum_{n=1}^{N}(P@n * rel(n))}{\#total\, relevant\, documents\, for\, this\, query}. \tag{4.3}$$

Berdasarkan contoh $P@n$, diketahui bahwa $P@n$ yang relevan adalah $P@1$, $P@2$, $P@3$, $P@5$, $P@7$, $P@8$, sehingga

$$AP = \frac{(\frac{1}{1} + \frac{2}{2} + \frac{3}{3} + \frac{4}{5} + \frac{5}{7} + \frac{6}{8})}{8}$$

MAP didapat dengan mencari nilai rata-rata AP untuk semua *query* [9].

4.2.3. Normalized Discount Cumulative Gain at Position n ($NDCG@n$). $NDCG$ merupakan ukuran evaluasi pada n dokumen teratas yang berasal dari hasil *ranking* menggunakan lebih dari dua label relevansi [6][9][11], formula $NDCG@n$ dituliskan sebagai berikut:

$$N(n) = Z_n \sum_{j=1}^{n} \frac{2^{r(j)} - 1}{\log_2(1 + j)} \tag{4.4}$$

Dimana $r(j)$ merupakan relevansi dari posisi dokumen ke j dari hasil *ranking*, dan Z_n merupakan normalisasi yang membuat *perfect rangking* $NDCG$ bernilai 1. Dari persamaan tersebut

$$2^{r(j)} - 1$$

disebut *gain* (G) dari posisi dokumen ke j,

$$\frac{(2^{r(j)} - 1)}{\log_2(1 + j)}$$

disebut *discounted gain* (DG),

$$\sum_{j=1}^{n} \frac{(2^{r(j)} - 1)}{\log_2(1 + j)}$$

disebut *discounted cumulative gain* (DCG) pada posisi ke n, dan terakhir

$$Z_n \sum_{j=1}^{n} \frac{(2^{r(j)} - 1)}{\log_2(1 + j)}$$

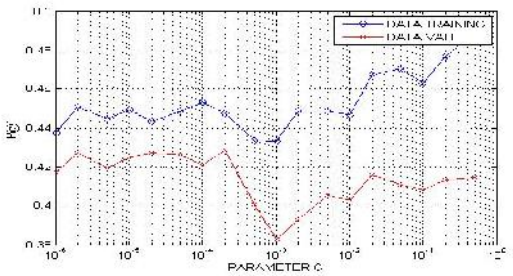
merupakan *Normalized Discounted Cumulative Gain* ($NDCG$) pada posisi ke n, yang disebut $NDCG@n$. Contoh menggunakan $NDCG@n$ dijelaskan pada Tabel 4.1.

TABEL 4.1
Contoh menghitung NDCG

Imperfect Ranking	**Formula**	
$(2, 1, 2, 1, 1, 0, 0)$		Label Relevansi : $2, 1, 0$
$(3, 1, 3, 1, 1, 0, 0)$	$2^{r(j)} - 1$	gains
$(1, 0.63, 0.5, 0.43, 0.39, 0.36, 0.33)$	$\frac{(1)}{\log_2(1+j)}$	Position discount
$(3, 3.63, 5.13, 5.56, 5.95, 5.95, 5.95)$	$\sum_{j=1}^{n} \frac{(2^{r(j)}-1)}{\log_2(1+j)}$	**Nilai DCG**
Perfect Ranking	**Formula**	
$(2, 2, 1, 1, 1, 0, 0)$		Label Relevansi : $2, 1, 0$
$(3, 3, 1, 1, 1, 0, 0)$	$2^{r(j)} - 1$	gains
$(1, 0.63, 0.5, 0.43, 0.39, 0.36, 0.33)$	$\frac{(1)}{\log_2(1+j)}$	Position discount
$(3, 4.89, 5.39, 5.82, 6.21, 6.21, 6.21)$	$\sum_{j=1}^{n} \frac{(2^{r(j)}-1)}{\log_2(1+j)}$	**Nilai Max DCG**
$(\mathbf{1}, \mathbf{0.74}, \mathbf{0.95}, \mathbf{0.96}, \mathbf{0.96}, \mathbf{0.96}, \mathbf{0.96})$	$\frac{DCG}{MaxDCG}$	**Nilai NDCG**

TABEL 4.2
Hasil five-fold cross validation pada data training

C	σ	P@1	P@3	P@5	MAP	$NDCG$@1	$NDCG$@3	$NDCG$@5
0.000001	0.1	0.4375	0.3736	0.3315	0.4570	0.3587	0.4021	0.4425
0.000002	0.1	0.4507	0.3876	0.3442	0.4747	0.3744	0.4211	0.4641
0.000005	0.1	0.4447	0.3842	0.3458	0.4737	0.3716	0.4199	0.4667
0.00001	0.1	0.4494	0.3883	0.3474	0.4756	0.3723	0.4219	0.4668
0.00002	0.1	0.4434	0.3831	0.3452	0.4715	0.3717	0.4154	0.4628
0.00005	0.1	0.4485	0.3865	0.3473	0.4751	0.3743	0.4213	0.4669
0.0001	0.1	0.4532	0.3870	0.3469	0.4772	0.3781	0.4220	0.4672
0.0002	0.1	0.4477	0.3836	0.3471	0.4760	0.3737	0.4191	0.4669
0.0005	0.1	0.4337	0.3621	0.3209	0.4477	0.3583	0.3933	0.4326
0.001	0.1	0.4333	0.3529	0.3100	0.4317	0.3607	0.3814	0.4164



GAMBAR 4.1. *Grafik P@1*

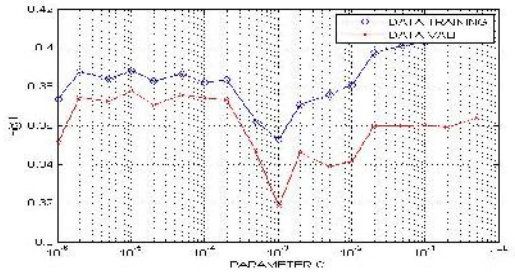
4.3. Hasil Eksperimen. Tabel 4.2 menunjukkan akurasi *ranking* metode SVR untuk dataset MQ2008 pada data *training*. Sedangkan, Tabel 4.3 menunjukkan akurasi *ranking* metode SVR untuk dataset MQ2008 pada data *validation*.

Dari beberapa parameter C dan $\sigma = 0.1$ yang di uji cobakan diperoleh kesimpulan sebagai berikut:

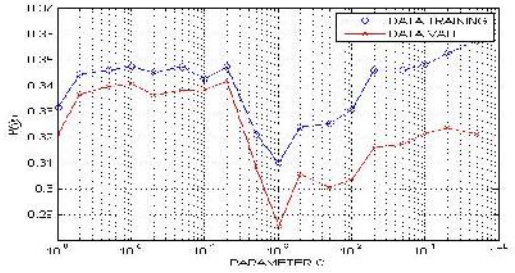
1. Pada Gambar 4.1 terlihat nilai P@1 terbaik pada data *training* diperoleh saat nilai $C = 0.0001$ dan $\sigma = 0.1$ dengan nilai 0.4532, sedangkan pada data *validation* diperoleh saat nilai $C = 0.0002$ dan $\sigma = 0.1$ dengan nilai 0.428524.
2. Pada Gambar 4.2 terlihat nilai P@3 terbaik pada data *training* diperoleh saat nilai $C = 0.00001$ dan $\sigma = 0.1$ dengan nilai 0.3883, sedangkan pada data *validation* diperoleh saat nilai $C = 0.00001$ dan $\sigma = 0.1$ dengan nilai 0.3780.
3. Pada Gambar 4.3 terlihat nilai P@5 terbaik pada data *training* diperoleh saat nilai $C = 0.00001$ dan $\sigma = 0.1$ dengan nilai 0.3474, sedangkan pada data *validation* diperoleh saat nilai $C = 0.0002$ dan $\sigma = 0.1$ dengan nilai 0.3418.
4. Pada Gambar 4.4 terlihat nilai MAP terbaik pada data *training* diperoleh saat nilai $C = 0.0001$ dan $\sigma = 0.1$ dengan nilai 0.4772, sedangkan pada data *validation* diperoleh saat nilai $C = 0.0002$ dan $\sigma = 0.1$ dengan nilai 0.4644.
5. Pada Gambar 4.5 terlihat nilai $NDCG$@1 terbaik pada data *training* diperoleh saat nilai $C =$

TABEL 4.3
Hasil five-fold cross validation pada data validation

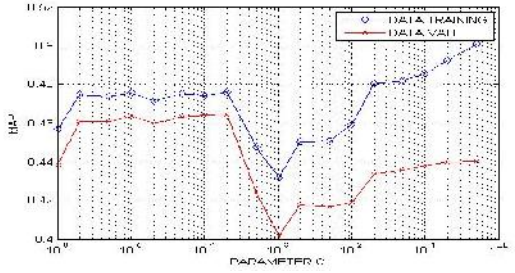
C	σ	P@1	P@3	P@5	MAP	$NDCG$@1	$NDCG$@3	$NDCG$@5
0.000001	0.1	0.4171	0.3512	0.3209	0.4385	0.3406	0.3790	0.4244
0.000002	0.1	0.4273	0.3746	0.3365	0.4611	0.3516	0.4069	0.4519
0.000005	0.1	0.4196	0.3724	0.3395	0.4610	0.3473	0.4123	0.4607
0.00001	0.1	0.4247	0.3780	0.3406	0.4635	0.3524	0.4140	0.4589
0.00002	0.1	0.4273	0.3703	0.3362	0.4597	0.3550	0.4073	0.4532
0.00005	0.1	0.4260	0.3758	0.3380	0.4634	0.3520	0.4119	0.4588
0.0001	0.1	0.4209	0.3741	0.3383	0.4641	0.3512	0.4107	0.4574
0.0002	0.1	0.4285	0.3733	0.3418	0.4644	0.3579	0.4117	0.4611
0.0005	0.1	0.4005	0.3461	0.3081	0.4240	0.3307	0.3705	0.4091
0.001	0.1	0.3827	0.3185	0.2855	0.4014	0.3180	0.3494	0.3827



GAMBAR 4.2. *Grafik P@3*



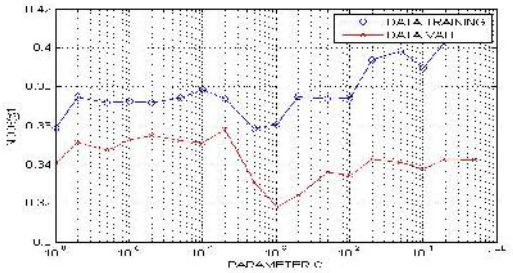
GAMBAR 4.3. *Grafik P@5*



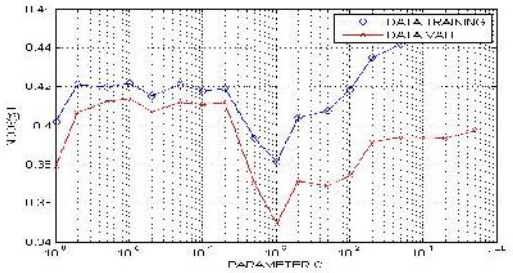
GAMBAR 4.4. *Grafik MAP*

0.0001 dan $\sigma = 0.1$ dengan nilai 0.3781, sedangkan pada data *validation* diperoleh saat nilai $C = 0.0002$ dan $\sigma = 0.1$ dengan nilai 0.3579.

6. Pada Gambar 4.6 terlihat nilai $NDCG$@3 terbaik pada data *training* diperoleh saat nilai $C = 0.0001$ dan $\sigma = 0.1$ dengan nilai 0.4220, sedangkan pada data *validation* diperoleh saat nilai $C = 0.00001$ dan $\sigma = 0.1$ dengan nilai 0.4140.
7. Pada Gambar 4.7 terlihat nilai $NDCG$@5 terbaik pada data *training* diperoleh saat nilai $C = 0.0001$ dan $\sigma = 0.1$ dengan nilai 0.4672, sedangkan pada data *validation* diperoleh saat nilai

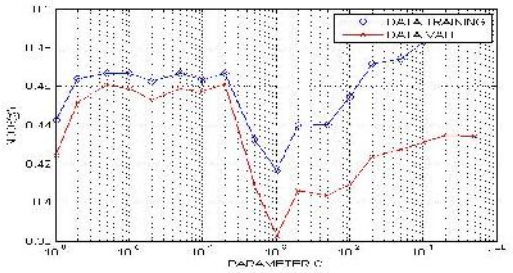


GAMBAR 4.5. *Grafik* $NDCG@1$



GAMBAR 4.6. *Grafik* $NDCG@3$

$C = 0.0002$ dan $\sigma = 0.1$ dengan nilai 0.4611.



GAMBAR 4.7. *Grafik* $NDCG@5$

Secara umum dari hasil uji coba pada data *training*, akurasi terbaik diperoleh dengan memilih parameter $C = 0.0001$ dan $\sigma = 0.1$. Sedangkan, hasil uji coba pada data *validation* akurasi terbaik diperoleh dengan memilih parameter $C = 0.0002$ dan $\sigma = 0.1$. Karena parameter $C = 0.0001$ dan $\sigma = 0.1$ mengalami penurunan peningkatan akurasi setelah di ujikan pada *training* maka parameter $C = 0.0001$ dan $\sigma = 0.1$ tidak bisa dijadikan parameter terbaik untuk di ujikan ke data *testing*, sedangkan parameter $C = 0.0002$ dan $\sigma = 0.1$ mengalami peningkatan saat diujikan pada data *validation* dan secara umum parameter $C = 0.0002$ dan $\sigma = 0.1$ merupakan parameter terbaik pada data *validation* sehingga parameter terbaik untuk diujikan pada data *testing* adalah parameter $C = 0.0002$ dan $\sigma = 0.1$. Hasil uji parameter $C = 0.0002$ dan $\sigma = 0.1$ pada data *testing* dapat dilihat pada Tabel 4.4.

5. Kesimpulan. Berdasarkan hasil eksperimen *learning to rank* dengan metode *Support Vector Regression* (SVR) yang telah dilakukan pada Dataset LETOR MQ2008, dengan parameter $C = 0.001$, 0.0005, 0.0002, 0.0001, 0.00005, 0.00002, 0.00001, 0.000005, 0.000002, 0.000001 dan $\sigma = 0.1$ yang dicoba, dapat ditarik kesimpulan sebagai berikut:

1. Kinerja *learning to rank* paling optimal diperoleh pada pemilihan parameter $C = 0.0002$ dan parameter $\sigma = 0.1$.
2. Kinerja *learning to rank* pada data *testing* dengan menggunakan parameter $C = 0.0002$ dan parameter $\sigma = 0.1$ memberikan hasil sebagai berikut: $P@1 = 0.422162$, $P@3 = 0.373289$, $P@5 =$

TABEL 4.4
Hasil uji parameter $C = 0.0002$ *dan* $\sigma = 0.1$ *pada data testing*

C	σ	$P@1$	$P@3$	$P@5$	MAP	$NDCG@1$	$NDCG@3$	$NDCG@5$
0.0002	0.1	0.4222	0.3733	0.3362	0.4620	0.3524	0.4140	0.4582

0.336227, $MAP = 0.462019$, $NDCG@1 = 0.352431$, $NDCG@3 = 0.414021$, $NDCG@5 = 0.458194$.

Kinerja *learning to rank* menggunakan metode SVR pada dataset LETOR MQ2008 dengan nilai parameter yang digunakan belum memperoleh hasil yang maksimal.

PUSTAKA

[1] BISHOP, C. M., *Pattern Recognition and Machine Learning*, Springer, 2006.

[2] CHAPELLE, O., CHANG, YI, *Yahoo! Learning to Rank Challenge Overview*, JMLR: Workshop and Conference Proceedings, 14 (2011), pp. 1-24.

[3] COSSOCK, D., ZHANG, T., *Subset Ranking Using Regression*, Proceedings of the 19th Annual Conference on Learning Theory (COLT 2006), (2006), pp. 605-619.

[4] CRISTIANNI, N. DAN SHAWE-TAYLOR, J., *An Introduction to Support Vector Machines*, Cambridge University Press, 2000.

[5] HAMEL, L., *Knowledge Discovery with Support Vector Machines*, New Jersey, John Wiley and Sons Inc, 2009.

[6] HANG LI., *Learning to Rank for Information Retrieval and Natural Language Processing*, Morgan and Claypool Publishers, 2011.

[7] JOACHIMS, T., *Optimizing Search Engines Using Clickthrough Data*, Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD), (2002), pp. 133-142.

[8] SCHOLKOPF, B DAN SMOLA, A., *Learning with Kernels*, The MIT Press, Cambridge, Massachusetts, 2002.

[9] TAO QIN,. JUN XU,. TIE-YAN LIU,. HANG LI,. , *LETOR: A Benchmark Collection for Research on Learning to Rank for Information Retrieval*, Inf Retrieval, 13 (2010), pp. 346-374.

[10] TIE-YAN LIU., *Learning to Rank for Information Retrieval*, Foundations and Trends in Information Retrieval, 3(3)(2009), pp. 225-331.

[11] TIE-YAN LIU., *Learning to Rank for Information Retrieval*, Springer, 2011.

[12] WITTEN, I. H., FRANK, EIBE DAN HALL, MARK A., *Data Mining Practical Machine Learning Tools and Techniques*, Burlington, USA, Morgan Kaufmann Publishers, 2011.