Content Retrieval dengan *FastText Word Embedding* pada *Learning Management System* Olimpiade

Rochana Prih Hastuti[1,*], Vellya Riona[1], Margareta Hardiyanti[1]
[1]Departemen Teknik Elektro dan Informatika, Sekolah Vokasi, Universitas Gadjah Mada
vellya.riona@mail.ugm.ac.id
margareta.hardiyanti@ugm.ac.id
[*]Korespondensi: rochana.prih.h@ugm.ac.id;

Abstract – Learning Management System (LMS) is a type of learning media that can be relied upon for students at various levels. Its use for competition purposes, specifically the Olympics, has its own characteristics compared to LMS for common learning purposes. One of them is the ability of the system to manage and retrieve the collection of problem sets which is relevant to users. Users on the Olympic LMS are segmented according to the field of science they want to be engaged in. Even so, each field has different types of learning topics and tends to grow over time. Pre process steps involving topic annotations needs experts and time-consuming. While to make search feature without using metadata information is of course also difficult to do. The semantic search feature becomes important in such a system. A content-based scheme is needed that is able to return problem sets relevant to the topic in each field. The search feature is built using an information retrieval scheme, namely the vector space model. The results of the experiment and the evaluation of the respondents showed that the word embedding representation with the best performance was the FastText word embedding. The efficiency of the model size is carried out by using the compressed version. This representation is not only able to accommodate the results according to the context of the query but also to solve problem of out-of-vocabulary.

Keywords – Information Retrieval, Vector Space Model, FastText Word Embedding

Intisari – *Learning Management System* (LMS) merupakan jenis media pembelajaran daring yang digunakan siswa di berbagai tingkat. Penggunaannya pada keperluan kompetisi, secara khusus olimpiade, memiliki karakteristik tersendiri dibanding LMS untuk keperluan pembelajaran sehari-hari. Salah satunya adalah kemampuan sistem mengelola bank soal dan menyajikan kategori yang relevan kepada *user*. *User* pada LMS Olimpiade tersegmentasi sesuai bidang ilmu yang ingin ditekuni. Meski begitu, tiap bidang memiliki topik pembelajaran yang beragam jenisnya dan bahkan cenderung berkembang seiring waktu. Manajemen bank soal dengan anotasi topik di awal memerlukan tenaga ahli dan memakan waktu. Sedangkan fitur pencarian tanpa menggunakan informasi metadata tersebut tentu juga sulit dilakukan. Fitur pencarian semantik menjadi penting pada sistem seperti ini. Dibutuhkan skema pencarian berdasarkan konten yang mampu mengembalikan soal-soal yang relevan dengan topik di masing-masing bidang. Fitur pencarian dibangun menggunakan skema *information retrieval* yakni *vector space model*. Hasil eksperimen dan evaluasi responden menunjukkan representasi *word embedding* dengan performa pencarian terbaik adalah FastText *word embedding*. Efisiensi ukuran model dilakukan dengan menggunakan *compressed version*. Representasi ini selain dapat mengakomodasi hasil pencarian sesuai konteks kueri juga dapat mengatasi permasalahan *out-of-vocabulary*.

Kata kunci – Information Retrieval, Vector Space Model, FastText Word Embedding

I. PENDAHULUAN

Penggunaan *Learning Management System* (LMS) sebagai media pembelajaran daring semakin meningkat di masa pandemi Covid-19. Siswa dari berbagai tingkat dasar, menengah, atas bahkan sampai mahasiswa menjadi sangat bergantung dengan keberadaan LMS. Penelitian tentang membangun *engagement* pada suatu LMS terus dilakukan untuk dijadikan rujukan pembangunan LMS yang lebih baik pada masa mendatang. Misalnya [1] dan [2] keduanya menganalisis bagaimana pengaruh penggunaan LMS dan bagaimana tantangan yang harus dipecahkan untuk membuat LMS yang lebih baik.

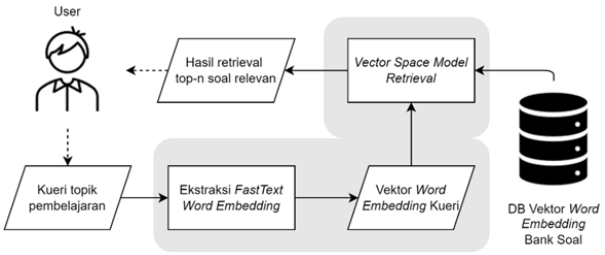
Di sisi lain, sistem serupa lain yang juga bisa dikategorikan sebagai LMS adalah sistem pembelajaran untuk domain kompetisi, misalnya berupa situs-situs https://www.hackerrank.com/ dan https://tlx.toki.id/. Situs-situs ini menyediakan berbagai persoalan untuk keperluan mengasah *skill* di bidang *competitive programming*. Jumlah soal yang disediakan terus bertambah, begitu pula topik bahasannya. Fitur penting yang dibutuhkan pada sistem seperti ini adalah *tagging* kategori soal-soal. Untuk sistem dengan enumerator yang memadai, maka metadata dapat dimanfaatkan untuk membangun indeks yang sesuai. Akan tetapi, untuk sistem lokal yang digunakan di sekolah-sekolah, di mana soal dikumpulkan secara sukarela, maka fitur *tagging* kategori akan sulit dibuat. Begitu pula dari sisi fitur pencarian, maka biasanya akan menggunakan metode *string matching* sederhana sebagai logika utamanya. Sistem informasi dengan menggunakan pendekatan *retrieval* banyak dibuat, misalnya adalah pada domain Al-Quran [3] [4], yang tentu memiliki jumlah data tekstual yang mencukupi. Kebanyakan dari metode yang digunakan pada penelitian tersebut mengandalkan algoritma *stemming* atau menggunakan korpus dan *thesaurus*.

Penggunaan algoritma-algoritma tersebut masih memiliki kekurangan berupa kueri *user* yang terbatas dengan kosakata yang ada pada korpus dan harus seragam. Selain itu, hasil pencarian yang diharapkan tentu adalah yang relevan dengan kueri yang diberikan. Oleh karenanya, pengembangan fitur pencarian dengan pendekatan *content retrieval* akan sangat membantu pada LMS dengan domain kompetisi seperti ini.

Penggunaan representasi fitur *FastText word embedding* akan menghasilkan representasi semantik yang akan menghasilkan hasil pencarian yang memiliki konten yang serupa dengan kueri yang diinputkan [5]. Selain itu, fitur *subword* dari *FastText,* juga dapat membantu mengatasi permasalahan *typo* dan *out-of-vocabulary*. Penggunaan *word embedding* ini didukung dengan adanya *pre-trained* model bahasa Indonesia yang dapat dengan mudah diimplementasikan [6].

II. METODOLOGI

Modul *content retreival* akan digunakan sebagai mesin utama dari fitur pencarian soal pada *Learning Management System* (LMS) Olimpiade berbasis web, yakni SOLVE. SOLVE memiliki fungsionalitas sebagai bank soal untuk beberapa kategori yang diunggah oleh *user*. Fitur pencarian sebelumnya menggunakan *string matching* digantikan oleh modul *content retrieval* berbasis *FastText word embedding*. Modul *content retrieval* akan dibuat dengan skema *Representational State Transfer Application Programming Interface* (REST API) tersusun dari beberapa metode terlihat pada Gambar 1 pada bagian berbayang abu-abu.



Gambar 1. Ilustrasi *searching* soal pada LMS dengan API *content retrieval*

A. Dataset

Data yang digunakan berasal dari bank soal aplikasi LMS SOLVE. Terdapat total 515 soal dari 9 mata pelajaran yang digunakan dalam eksperimen penelitian ini terlihat pada Tabel 1. Jumlah ini cukup terbatas dikarenakan aplikasi tersebut masih dalam fase awal rilis, sehingga pengguna yang melakukan penyimpanan soal juga terbatas.

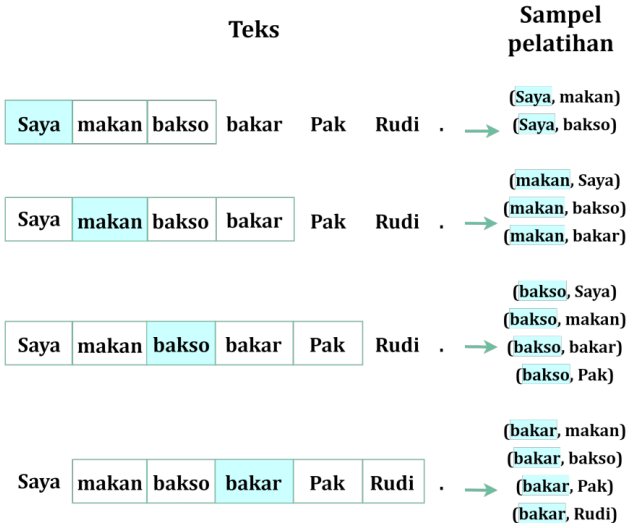
Tabel 1. Jumlah soal tiap mata pelajaran

No	Mata pelajaran	Jumlah soal
1	Informatika	125
2	Biologi	87
3	Kebumian	50
4	Geografi	117
5	Astronomi	59
6	Ekonomi	16
7	Kimia	20
8	Matematika	21
9	Fisika	20

B. *FastText Word Embedding*

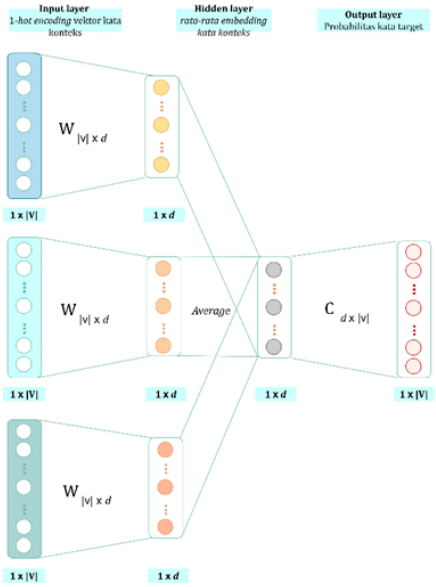
FastText merupakan metode ekstraksi fitur dari kata ke dalam bentuk bilangan rill dengan konsep *prediction-based word embedding* [5]. *FastText* adalah pengembangan dari metode continuous *bag-of-words* (CBOW) [7], dengan skema input *one-hot encoding* dari kata konteks dan menghasilkan output *one-hot encoding* dari kata target.

Pengambilan kata konteks dilakukan dengan menggunakan *sliding window* dengan ukuran tertentu. Setiap kata yang berada dalam *window* dari kata target akan menjadi kata konteks. Berikut merupakan contoh *window* dengan ukuran 2 untuk teks "Saya makan bakso bakar Pak Rudi." yang dapat dilihat pada Gambar 2. Kata yang berbayang biru adalah kata target. Dalam satu sampel, sebuah kata target dapat memiliki beberapa kata konteks.



Gambar 2. Kata konteks dari window berukuran 2

Arsitektur CBOW terdiri dari tiga *layer*, yaitu *layer input* dengan neuron sejumlah banyak *unique vocabulary* $\|V\|$ dari data pelatihan, *hidden layer* dengan neuron sejumlah dimensi fitur yang diinginkan d, dan *layer output* dengan ukuran $\|V\|$. Layer input akan menerima vektor *one-hot encoding* dari kata konteks, sedangkan *layer output* dilatih agar dapat menghasilkan vektor *one-hot encoding* dari kata target. Setiap kata target dapat memiliki lebih dari satu kata konteks dalam satu sampel. Sehingga, untuk menghitung nilai pada *hidden layer*, dihitung terlebih dahulu rata-rata hasil *layer* sebelumnya dari semua kata konteks dalam satu sampel. Vektor fitur yang akan digunakan dari proses pelatihan CBOW ini adalah nilai hasil neuron pada bagian *hidden layer* CBOW. Ilustrasi ditampilkan pada Gambar 3.



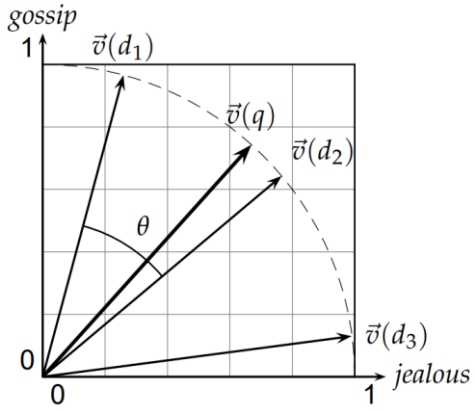
Gambar 3. Arsitektur skema CBOW secara umum
Sumber: berasal dari [8]

Perbedaan *FastText* terletak pada *input* berupa *character n-gram subword*, sehingga ekstraksi fitur dapat dilakukan pada level prefiks dan sufiks di luar data pelatihan atau *out-of-vocabulary* (OOV). Pada penelitian ini digunakan model *pre-trained FastText* Bahasa Indonesia dengan skema CBOW [6].

Akan tetapi, ukuran model *FastText* berkisar antara 3-7 Gb, sehingga sulit untuk diproses dengan spesifikasi RAM pada server biasa. *Compressed FastText* adalah pendekatan untuk mengurangi ukuran model *FastText* dengan beberapa proses, yakni penghilangan *negative-vector* untuk *extend* pelatihan, pengurangan *matrix vocabulary*, dan penentuan ukuran matriks baru yang efisien dengan memperhatikan *collision uniformity* [9]. Ukuran model yang dihasilkan berkurang jauh yakni pada kisaran 15-20 Mb, sehingga lebih *feasible* untuk digunakan pada tahap *deployment* aplikasi.

C. *Vector Space Model*

Representasi sekumpulan dokumen, dalam hal ini bank soal, sebagai vektor pada sebuah ruang vektor disebut dengan *vector space model* [10]. Kueri dapat direpresentasikan sebagai vektor pada ruang vektor yang sama. Misalkan vektor sebuah dokumen d dinotasikan dengan $\vec{V}(d)$, yakni vektor *FastText sentence embedding* dengan dimensi tertentu. Dua vektor, vektor dokumen dan vektor kueri, terlepas dari ukuran *vocabulary*-nya dapat diukur tingkat kemiripannya dengan melihat besarnya sudut yang terbentuk. *Cosine similarity* adalah besarnya sudut yang dibentuk dari kedua vektor tersebut terlihat pada Gambar 4 dapat dihitung dengan rumus (1).



Gambar 4. Ilustrasi *cosine similarity*. $sim(d_1, d_2) = \cos\theta$

Sumber: berasal dari [10]

Sebelum menggunakan besarnya sudut untuk kuantifikasi kemiripan, perlu dihitung *magnitude* dari kedua vektor. Akan tetapi, pendekatan ini lemah jika panjang vektor dari kedua dokumen signifikan berbeda, meskipun konten yang dimiliki mirip. Ini berarti meskipun distribusi relatif kedua dokumen serupa, namun nilai absolut dari frekuensi *vocab* yang dimiliki berbeda jauh. Untuk menormalkan hal ini, cara standar mengkuantifikasi similaritas dari dua dokumen yang direpresentasikan dengan vektor numerik adalah dengan menghitung *cosine similarity*-nya. Efek dari pembagian dengan penyebut pada (1) adalah untuk menormalkan vektor $\vec{V}(d_1)$ dan $\vec{V}(d_2)$ menjadi unit vektor $\vec{v}(d_1) = \vec{V}(d_1)/|\vec{V}(d_1)|$ dan $\vec{v}(d_2) = \vec{V}(d_2)/|\vec{V}(d_2)|$.

$$sim(d_1, d_2) = \frac{\vec{V}(d_1).\vec{V}(d_2)}{|\vec{V}(d_1)||\vec{V}(d_2)|} \quad (1)$$

Sehingga, (1) dapat ditulis ulang menjadi (2), yang merupakan *dot product* dari kedua vektor dokumen yang sudah dinormalkan.

$$sim(d_1, d_2) = \vec{v}(d_1).\vec{v}(d_2) \quad (2)$$

Perhitungan ini adalah *cosine* dari sudut θ yang diapit kedua vektor pada Gambar 4. Nilai *similarity* ini dapat digunakan jika sistem memiliki sekumpulan dokumen dalam hal ini bank soal, maka kueri yang juga merupakan dokumen d dapat dicari nilai similaritas tertingginya dengan dokumen lain dalam bank soal. Pencarian seperti ini bermanfaat jika *user* mencari tidak hanya satu dokumen, melainkan beberapa dokumen lain dengan nilai similaritas tertinggi. Sehingga, perhitungan *dot product* antara $\vec{v}(d)$ dihitung terhadap setiap dokumen dalam koleksi $\vec{v}(d_1), \ldots, \vec{v}(N)$, kemudian pilih beberapa dokumen dengan nilai similaritas tertinggi sesuai yang dikehendaki.

Tabel 2. Hasil relevansi *precision* terhadap *groundtruth* dokumen

No	Skema kueri	Compressed FastText	SQL
1	1 kata	0.32	0.28
2	2 kata	0.28	0.16
3	1 kata *typo*	0.12	0
4	2 kata *typo*	0.11	0
5	1 kata OOV	0.15	0
6	2 kata OOV	0.13	0

Tabel 3. Hasil relevansi *precision* terhadap penilaian responden

No	Skema kueri	Compressed FastText	SQL
1	1 kata	0.5	0.32
2	2 kata	0.48	0.17
3	1 kata *typo*	0.19	0
4	2 kata *typo*	0.20	0
5	1 kata OOV	0.25	0
6	2 kata OOV	0.30	0

D. Evaluasi

Pengukuran efektivitas sebuah sistem *retrieval* memerlukan beberapa komponen yakni koleksi dokumen, kueri, dan standar pengujian [10]. Fokus utama pengujian adalah relevansi dokumen yang dihasilkan untuk tiap kueri.

1. Pengujian
   Tingkat relevansi pada penelitian ini diukur menggunakan dua jenis acuan, kategori asal dokumen kemudian disebut relevansi *groundtruth* dan pengujian oleh pengguna sistem selanjutnya disebut relevansi responden. Pencarian sistem dilakukan pada seluruh dokumen di bank soal yakni sejumlah 515 dokumen.
2. Kueri
   Seluruh kategori dokumen terdiri dari 9 mata pelajaran akan diuji dengan beberapa kueri bertipe kata dan frasa (dua kata), serta tiga jenis kueri yakni standar, *typo*, dan *out-of-vocabulary* (OOV). Sehingga, total diujikan 12 kueri untuk tiap kategori dokumen. Jumlah dokumen yang dihasilkan 10 buah, menyesuaikan ukuran terkecil dari kategori mata pelajaran yakni 20, agar perhitungan dengan matriks evaluasi tidak rancu karena mengembalikan terlalu banyak dokumen.
3. Matriks evaluasi
   Matriks evaluasi yang digunakan adalah nilai *precision*. Nilai ini digunakan untuk mengukur efektivitas sistem pencarian dalam menampilkan konten paling relevan pada halaman pertama, yang menjadi tipikal pengguna sistem pencarian. *Precision* (P) adalah bagian dari hasil pencarian yang dinilai relevan, dihitung dengan (3).

$$Precision = \frac{\#(relevant\ items\ retrieved)}{\#(retrieved\ items)} \quad (3)$$

III. HASIL DAN PEMBAHASAN

Hasil *precision* untuk pengujian dengan *groundtruth* dokumen dan penilaian responden berturut-turut ditampilkan pada Tabel 2 dan Tabel 3. Setiap jenis kueri diujikan untuk seluruh 9 mata pelajaran masing-masing dengan dua kueri. Nilai merupakan rata-rata persentase dari total 10 dokumen yang dikembalikan untuk tiap kueri. Tampak pada kedua tabel *Compressed FasText* memiliki nilai *precision* lebih tinggi dibanding kueri menggunakan SQL biasa.

Tabel 4. Kueri dan hasil pencarian kategori Informatika

No	Skema kueri	Kueri	Compressed FastText
1	1 kata *typo*	mthod	*Method overriding* berbeda dengan *method*…
2	2 kata *typo*	kywrd fnal	*Keyword this* dapat digunakan untuk membedakan *variable*…
3	1 kata OOV	undefined	*Method Accessor* tidak menggunakan parameter…
4	2 kata OOV	switch case	Konstruktor pada suatu *class* boleh menggunakan…

Contoh hasil pencarian untuk beberapa kueri sampel dari kategori Informatika dapat dilihat pada Tabel 4. Contoh nomor 1 dan 2 adalah dua contoh kata yang mengalami *typo* atau kesalahan penulisan (seharusnya "method" namun ditulis "mthod" pada kueri). Sedangkan contoh nomor 3 dan 4 adalah kueri yang kosakatanya tidak ada pada data.

Pada Tabel 4, terlihat bahwa model representasi vektor dengan *Compressed FasText word embedding*, dapat mengembalikan hasil pencarian dengan konten yang sesuai dengan kueri yang diujikan meskipun menggunakan kueri bertipe *typo* maupun *out-of-vocabulary*. Hal ini sesuai dengan dugaan awal bahwa representasi vektor dengan skema *word embedding* memuat konteks dan dapat mengembalikan hasil pencarian dengan lebih baik. Selain itu, representasi ini juga dapat mengatasi permasalahan *out-of-vocabulary*. Pencarian menggunakan kueri yang sama pada SQL tidak akan memberikan hasil.

IV. SIMPULAN

Modul *content retrieval* telah dibuat dan diintegrasikan sebagai opsi fitur pencarian di aplikasi LMS Olimpiade SOLVE. Performa model berbasis *FastText word embedding* mengungguli metode awal *string matching* pada dua acuan evaluasi: *automatic* (*groundtruth*) dan *human evaluation* (responden). Penggunaan *Compressed FastText word embedding* mengatasi permasalahan ukuran model, sehingga modul siap untuk digunakan di tahap produksi.

REFERENSI

[1] S. A. Raza, W. Qazi, K. A. Khan, and J. Salam, "Social isolation and acceptance of the learning management system (LMS) in the time of COVID-19 pandemic: an expansion of the UTAUT model," *J. Educ. Comput. Res.*, vol. 59, no. 2, pp. 183–208, 2021.

[2] U. Alturki and A. Aldraiweesh, "Application of learning management system (Lms) during the covid-19 pandemic: A sustainable acceptance model of the expansion technology approach," *Sustainability*, vol. 13, no. 19, p. 10991, 2021.

[3] I. Humaini, T. Yusnitasari, L. Wulandari, D. Ikasari, and H. Dutt, "Informatian Retrieval of Indonesian Translated version of Al Quran and Hadith Bukhori Muslim," in *2018 International Conference on Sustainable Energy, Electronics, and Computing Systems (SEEMS)*, IEEE, 2018, pp. 1–5.

[4] A. Aulia, D. Khairani, and N. Hakiem, "Development of a retrieval system for Al Hadith in Bahasa (case study: Hadith Bukhari)," in *2017 5th International Conference on Cyber and IT Service Management (CITSM)*, IEEE, 2017, pp. 1–5.

[5] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Trans. Assoc. Comput. Linguist.*, vol. 5, pp. 135–146, 2017.

[6] E. Grave, P. Bojanowski, P. Gupta, A. Joulin, and T. Mikolov, "Learning word vectors for 157 languages," *ArXiv Prepr. ArXiv180206893*, 2018.

[7] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *ArXiv Prepr. ArXiv13013781*, 2013.

[8] R. P. Hastuti, "Q-Learning untuk Pembentukan Tree dalam Sentence Generation Bahasa Indonesia dengan Algoritme Tree Long Short-Term Memory," PhD Thesis, Universitas Gadjah Mada, 2020.

[9] D. Dale, "Compress-fastText." May 06, 2023. Accessed: May 24, 2023. [Online]. Available: https://github.com/avidale/compress-fasttext

[10] H. Schutze, C. D. Manning, and P. Raghavan, *Introduction to information retrieval*. Cambridge University Press, 2008.