Sistem Rekomendasi Podcast Menggunakan Metode N-Gram dan Term Frequency

Vincent Theonardo[1], Jimmy Lohil[2], Goldwin Chantesuta[3], Wenripin Chandra[4], Arwin Halim*[5]
STMIK Mikroskil, Jl. Thamrin No. 112, 124, 140, Telp. (061) 4573767, Fax. (061) 4567789
Program Studi Teknik Informatika, STMIK Mikroskil, Medan
[1]151111747@students.mikroskil.ac.id, [2]151111828@students.mikroskil.ac.id,
[3]151110521@students.mikroskil.ac.id, [4]wenripin@mikroskil.ac.id, [5]arwin@mikroskil.ac.id

Abstrak

Salah satu bentuk utama penyajian informasi dan hiburan adalah dalam bentuk konten audio on-demand atau yang sering disebut sebagai podcast. Permasalahan yang dihadapi adalah kesulitan pengguna untuk menemukan podcast yang sesuai dengan preferensinya di antara ratusan ribu podcast yang kini tersedia di internet. Selain jumlah podcast, tidak lengkapnya metadata untuk podcast menjadi salah satu masalah dalam mengembangkan sistem rekomendasi pada umumnya. Sistem ini dirancang dengan memanfaatkan N-Gram dan Term Frequency untuk menghasilkan kueri dari judul dan deskripsi sebuah podcast yang kemudian akan digunakan untuk mencari podcast lainnya yang mirip dengan memanfaatkan perhitungan cosine similarity. Tahap pengujian dilakukan melalui pengujian terhadap sistem menggunakan perhitungan nDCG untuk menentukan tingkat relevansi hasil rekomendasi. Dari hasil pengujian terhadap sistem, nilai rata-rata tingkat relevansi rekomendasi podcast adalah 53.8% dan rata-rata f1-score terbaik dari 10 kategori terbaik adalah 14%.

Kata kunci*—Cosine Similarity, N-Gram, Sistem Rekomendasi Padcast, Term Frequency*

Abstract

One of the main forms of information and entertainment presentation is in the form of on-demand audio content or often referred as podcast. The problem met is the difficulty of users to find podcasts that match their preferences among hundreds of thousands of podcasts that are now available on the internet. Besides the number of podcasts, the incompletion of podcasts metadata is one of the problems in developing a recommendation system in general.This system is designed by utilizing N-Gram and Term Frequency to generate queries from the title and description of a podcast which will then be used to find other podcasts that are similar by utilizing Cosine Similarity calculations. The testing phase is done through by testing the system using nDCG calculations to determine the relevance level of the recommendation results. From the results, the average value of the relevance level of podcast recommendations is 53.8% and the best average f1-score of the 10 best categories is 14%.

Keywords*—Cosine Similarity, N-Gram, Podcast Recommendation System, Term Frequency*

1. PENDAHULUAN

Podcast merupakan konten audio *on-demand* yang telah berkembang menjadi salah satu *channel* informasi, hiburan dan iklan [1]. Diestimasikan ada 90 juta *user* di Amerika Serikat pernah mendengar *podcast* setiap bulannya, naik 20 juta atau 6% dari tahun 2018 [2]. Pada April 2018, Apple menginformasikan bahwa ada 525.000 *podcast* dengan 18,5 juta episode yang tersedia di iTunes [3]. Karena jumlah dan heterogenitas konten *podcast* yang besar, menemukan *podcast* yan sesuai dengan preferensi merupakan hal yang menantang. Mengkategorikan atau memberikan *tag* pada *podcast* sehingga dapat dimanfaatkan pada sistem rekomendasi pada umunya dalam menghasilkan *podcast* sesuai preferensi *user* juga merupakan hal yang sulit untuk dilakukan sebagai akibat dari metadata seperti ID3 *tags* (*genre*, artis, dll) yang tidak lengkap [4].

Sistem rekomendasi *podcast* yang ada pada saat ini menerapkan metode *Natural Language Processing* (NLP) dengan memanfaatkan *Conditional Random Field* (CRF) *tokenizer* dan *Skip-gram*

Negative Sampling (SGNS) untuk mempelajari fundamental kata dan hubungan antar kata yang terdapat pada deskripsi *podcast.* Namun metode yang diajukan hanya dapat digunakan pada *podcast* berbahasa Mandarin [4]. Adapaun beberapa layanan *music streaming* yang menyediakan *podcast* seperti Spotify[1], Soundcloud[2], maupun Google Podcast[3] tidak menyediakan fitur rekomendasi *podcast* yang sesuai dengan preferensi *user* dan mengharuskan *user* mencari *podcast* melalui fitur *search* atau kategori. Podible[4], salah satu *website* penyedia *podcast* memberikan fitur pencarian yang cukup baik dan rekomendasi yang berdasarkan pada *history user.* Namun Podible hanya beroperasi pada platform *web* dan iOS.

Walaupun dengan terbatasnya metadata *podcast,* sistem rekomendasi *podcast* dapat dikembangkan dengan menggunakan metode *n-gram* dan *term frequency.* Nascimento,dkk memanfaatkan metode *n-gram* dan *term frequency* untuk menghasilkan sistem rekomendasi makalah penelitian walaupun dengan keterbatasan metadata. Dengan memanfaatkan metadata judul dan abstrak, metode *n-gram* dan *term frequency* digunakan untuk menghasilkan *query* makalah penelitian yang akan dijadikan sebagai *input* pada sistem rekomendasi. Metode *n-gram* dan *term frequency* juga menunjukkan hasil yang lebih baik dibandingkan dengan metode *n-gram & term position, noun phrase & term frequency* dan *noun phrase & term position* yang juga dapat digunakan untuk menghasilkan *query.* Selain pemanfaatan *n-gram* dan *term frequency,* rekomendasi akan dihasilkan melalui pemanfaatan algoritma rekomendasi *content-based* [5].

2. METODE PENELITIAN

Metode yang digunakan dalam penelitian ini adalah metode rekomendasi dengan memanfaatkan *n-gram & term frequency.* Proses dari sistem rekomendasi ini akan terbagi menjadi 6 langkah secara keseluruhan dimana langkah pertama merupakan pengumpulan data, *pre-processing*, 2 langkah berikutnya merupakan tahap untuk menghasilkan kandidat *podcast,* dan 3 tahap berikutnya untuk menghasilkan rekomendasi. Langkah berikutnya adalah melakukan pengujian dengan menghitung nilai *precision, recall, f1 score*, dan nDCG dari hasil rekomendasi yang didapat. Gambar 1 menunjukkan diagram alir dari keseluruhan proses rekomendasi *podcast* dengan *n-gram* dan *term frequency*.

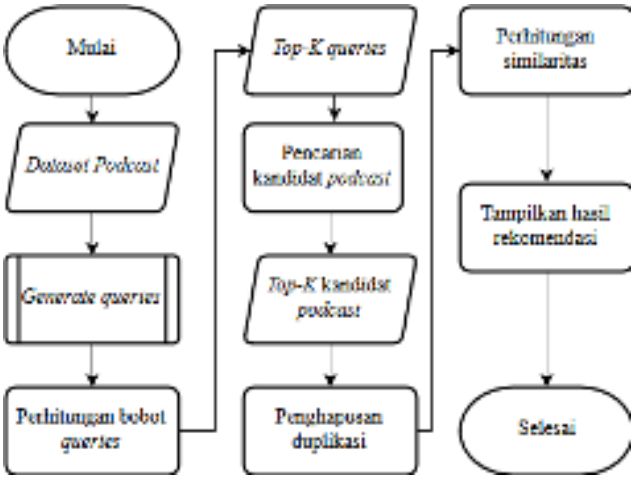
2.1 Dataset Podcast

Dataset yang digunakan merupakan dataset yang dibagikan di Kaggle dengan rincian:

a. Episodes.csv dengan 881.046 episode acak
b. podcasts.csv yang terdiri dari:
    i. 118.313 judul unik pada *podcast*,
    ii. 43 bahasa pada *podcast* dan 82% berbahasa inggris,
    iii. 18800 kategori unik pada *podcast*.

Untuk menghasilkan sistem rekomendasi, informasi dataset *podcast* yang digunakan adalah judul dan deskripsi dalam bahasa Inggris dan huruf latin pada *podcast*.

[1] https://www.spotify.com
[2] https://soundcloud.com
[3] https://play.google.com/store/apps/details?id=com.google.android.apps.podcasts&hl=en
[4] https://podible.co



Gambar 1 *Flowchart* Sistem Rekomendasi *Podcast*

2.2 Generate Queries

Terdapat 4 tahapan yang harus dilakukan untuk menghasilkan *query* yaitu :

1. Transformasi menjadi *lowercase*
   Tahap pertama yang dilakukan adalah mengubah huruf kapital pada teks menjadi huruf kecil. Berikut hasil teks judul dan deskripsi yang telah dilakukan transformasi :
   Judul : *good food eating* : *nutrition and diet* | *natural weight loss* | *healthy lifestyle*
   Deskripsi : *focusing on real food nutrition and sharing empowering diet information so you can live a healthy lifestyle and most importantly be happy*
2. Penghapusan *stopwords* dan tanda baca
   Penghapusan *stopwords* dan tanda baca dilakukan untuk menghindari kata yang tidak bermakna. Berikut hasil teks judul dan deskripsi yang telah dilakukan penghapusan *stopwords* dan tanda baca:
   Judul : *good food eating nutrition diet natural weight loss healthy lifestyle*
   Deskripsi : *focusing real food nutrition sharing empowering diet information so you can live a healthy lifestyle and most importantly be happy*.
3. *Stemming*
   Proses *stemming* dilakukan untuk mencegah terjadinya duplikasi *query* yang mirip. Salah satu contoh metode *stemming* yang banyak digunakan adalah *Porter Stemming* [6]. *Porter Stemming* merupakan proses penghilangan sufiks pada kata-kata dalam bahasa Inggris sehingga didapatkan bentuk dasar dari sebuah kata. Dengan mengabaikan bentuk asal suatu kata, dapat dikatakan sebuah dokumen merupakan bentuk representasi vektor kata-kata atau istilah-istilah. Sebuah istilah yang memiliki bentuk dasar yang sama cenderung memiliki arti yang sama, contohnya :
   CONNECT
   CONNECTED
   CONNECTING
   CONNECTION
   CONNECTIONS
   Seringkali performa pada sistem IR (*Information Retrieval*) dapat ditingkatkan jika istilah-istilah seperti diatas disatukan menjadi satu istilah. Hal ini dapat dilakukan dengan menghilangkan beberapa sufiks seperti -ED, -ING, -ION, -IONS sehingga didapatkan bentuk dasar berupa istilah *CONNECT*. Sebagai tambahan, proses menghilangkan sufiks akan mengurangi jumlah istilah yang terdapat pada sistem IR, yang berakibat pada berkurangnya ukuran dan kompleksitas data [7].
4. *N-Gram*
   Sebuah *N-gram* adalah N buah karakter yang dipotong dari *string* yang lebih panjang [8]. *N-gram* sendiri dapat tersusun dari jumlah kata yang berbeda seperti [9] :
   a. *Unigram* : *n-gram* berukuran 1

b. *Bigram* : *n-gram* berukuran 2
c. *Trigram* : *n-gram* berukuran 3
d. *Quadgram* : *n-gram* berukuran 4
e. Dst

Secara singkat *N-gram* adalah metode untuk menghasilkan *query* dengan mengekstraksi 'n' kata berikutnya dari teks *input* [5]. *N-gram* juga dapat berupa gabungan beberapa *items* yang berasal dari sebuah rangkaian yang dimasukkan. *Items* disini dapat berupa [9] :

1. Fonem
2. Silabel
3. Huruf
4. Kata
5. Bentuk apa saja tergantung aplikasi

Cara kerja *n-gram* dapat dilihat melalui contoh berikut:
Input : "*the dog smelled like a skunk*"

1. Pada bentuk *bigram* : "*# the*", "*the dog*", "*dog smelled*", "*smelled like*", "*like a*", "*a skunk*", "*skunk #*"
2. Pada bentuk *trigram* : "*# the dog*", "*the dog smelled*", "*dog smelled like*", "*smelled like a*", "*like a skunk*", "*a skunk #*"

Model *n-gram* banyak digunakan untuk pekerjaan seperti mengidentifikasi bahasa secara statistik, mengoreksi pengejaan. *Low-order word n-gram models* juga digunakan untuk pemodelan bahasa pada teknologi *speech recognition* [10]. Dengan menentukan ukuran jendela dan menggeser jendela ini ke atas segmen teks. Setiap kata mengambil satu tempat di jendela. Dengan demikian, *query* yang dihasilkan dengan mengambil istilah di dalam jendela [5].

Perkembangan pada NLP telah mencatatkan tingkat kesuksesan yang tinggi pada beberapa jenis tantangan. NLP memungkinkan kita untuk [11] :

1. Mengklasifikasikan kata-kata ke dalam kategori menurut tata bahasa / gramatikal (contoh : kata benda, kata kerja).
2. Menentukan makna dari kata yang memiliki beberapa makna sekaligus, dengan berdasarkan konten dari sebuah dokumen.
3. Menguraikan kalimat, yaitu dengan melakukan analisa terhadap tata bahas. Hal ini memungkinkan kita untuk menghasilkan representasi struktur tata bahasa yang utuh.

Pada tahapan ini, data linguistik akan diekstrak dari dokumen yang mempunyai data yang belum terstruktur [11].

2.3 Perhitungan Bobot Kueri

Bobot suatu *query* akan dihitung berdasarkan frekuensi *query* muncul dalam teks, maka dapat dihitung dengan menggunakan persamaan 1.

$$w(q) = \gamma\, x\, \frac{freq(q)}{\text{max\_frequency}} + (1-\gamma) x\, \frac{word\_frequency(q)}{\text{max\_}word} \quad (1)$$

Keterangan :
w = bobot
q = *query*
γ = tingkat kepentingan *query*
freq(q) = frekuensi *query* muncul dalam teks
max_frequency = frekuensi tertinggi di antara semua frekuensi
max_word = nilai tertinggi yang dihasilkan dari persaman (2)

Untuk menghitung bobot suatu *query,* terlebih dahulu untuk menghitung frekuensi tertinggi *query* muncul dalam teks dengan menggunakan persamaan 2.

$$word\_frequency(q) = \sum_k \frac{freq(i_k)}{\text{max\_}word\_frequency} \quad (2)$$

Keterangan :
max_word_frequency = frekuensi tertinggi *query* muncul di teks
i_k = kata keberapa di dalam *query*

Sehingga, istilah yang paling sering muncul dalam sebuah dokumen akan mendapatkan TF sebesar 1 dan istilah lainnya akan mendapatkan pecahan sebagai *term frequency* dari dokumen tersebut [12].

2.4 Pencarian Kandidat *Podcast*

Dengan memanfaatkan dataset yang disediakan oleh ListenNotes.com [5], 5 *query* dengan bobot tertinggi diperlakukan sebagai input. Dari setiap *query,* akan dihasilkan 20 kandidat *podcast.* Jumlah kandidat *podcast* yang akan dihasilkan adalah 5 *query* x 20 kandidat *podcast* = 100 kandidat *podcast*. Dari 100 *podcast* yang dihasilkan, akan dilakukan proses pencocokan dengan *dataset.* Apabila tidak terdapat dalam *dataset* yang digunakan, maka kandidat *podcast* akan dihapus.

2.5 Penghapusan Duplikasi

Kandidat *podcast* akan dicari duplikasinya dan dilakukan penghapusan sehingga sistem rekomendasi akan bekerja lebih maksimal. Penghapusan duplikasi dilakukan dengan cara mengubah setiap judul *podcast* menjadi *lower-case* lalu dilanjutkan dengan pencocokan *string* pada judul.

2.6 Perhitungan Similaritas

Metode *similarity* digunakan untuk mencari dokumen paling mirip dengan menggunakan informasi tekstual. Metode *cosine similarity* digunakan dengan cara mendokumentasikan kesamaan metriks. Dengan metriks kesamaan *cosine*, dokumen direpresentasikan sebagai vektor dimana setiap posisi dalam vektor mewakili frekuensi suatu istilah dalam dokumen [13]. Pada metode ini *angle* yang terbentuk antara dua buah vektor yang diuji, merupakan ukuran perbedaan antara kedua vektor tersebut, dan kosinus dari sudut tersebut digunakan sebagai ukuran kemiripan numerik (karena kosinus memiliki properti unik, dimana ia akan bernilai 1 untuk vektor yang identik dan 0 untuk vektor orthogonal) [14].

Untuk menghitung kemiripan dua dokumen berdasarkan bobot *query* dihitung dengan persamaan 3.

$$Cosine(i,j) = \frac{\sum_k W_{ik} W_{jk}}{\left(\sum_k W_{ik}^2 \; x \; \sum_k W_{jk}^2\right)^{1/2}} \quad (3)$$

Keterangan :
W_{ik} = bobot istilah k di dokumen i
W_{jk} = bobot *query* ke-k di dokumen j
$\sum k$ = penjumlahan sebanyak k kali

Berdasarkan nilai kemiripan yang diperoleh dari persamaan diatas kemudian akan dilakukan pengurutan secara menurun dari nilai kemiripan yang paling tinggi ke rendah [13].

2.7 Evaluasi Hasil Rekomendasi

Dari daftar *podcast* yang telah dihitung kemiripannya berdasarkan *cosine similarity*, *podcast* tersebut kemudian diurutkan berdasarkan nilai kemiripannya. Pada implementasinya rekomendasi yang akan ditampilkan kepada pengguna diambil dari 20 *podcast* teratas pada daftar urutan tersebut. Dari hasil rekomendasi akan dievaluasi menggunakan nilai presisi, recall, f1-score dan nDCG.

2.7.1 Precision

Precision melambangkan proporsi dari *Predicted Positive* yang merupakan *Real Positives* [15]. Secara singkat, *Precision* merupakan pecahan dari dokumen yang didapatkan yang bersifat relevan [16]. Perhitungan untuk mencari *Precision* dapat dilihat pada persamaan 4.

[5] https://www.listennotes.com/api/

$$Precision = \frac{\#(relevant\ items\ received)}{\#(retrieved\ items)} = P(relevant|retrieved) \quad (4)$$

2.7.2 Recall

Recall atau disebut juga sensitivitas adalah bagian dari *Real Positive* yang merupakan *Predicted Positive* yang benar [15]. Secara singkat, Recall merupakan pecahan dari dokumen relevan yang didapat. Persamaan 5 menampilkan perhitungan untuk mencari Recall.

$$Recall = \frac{\#(relevant\ items\ retrieved)}{\#(relevant\ items)} = P(retrieved|relevant) \quad (5)$$

2.7.3 F1-Score

F1 Score merupakan perhitungan tingkat akurasi dengan memanfaatkan *precision* dan *recall*. Nilai tertinggi pada *F1 score* adalah 1 (nilai sempurna pada *precision* dan *recall*) dan nilai terendahnya adalah 0. Persamaan 6 menampilkan perhitungan untuk mencari nilai F1 Score.

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \quad (6)$$

2.7.4 Evaluasi Hasil Rekomendasi

Normalized Discounted Cumulative Gain (NDCG) merupakan DCG yang telah dinormalisasi. NDCG dapat dikalkulasikan dengan persamaan 7.

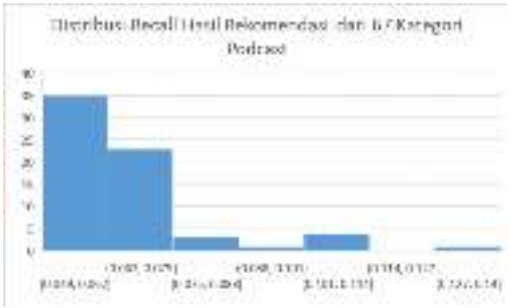
$$N\ DCG_p = \frac{DCG_p}{I\ DCG_p} \quad (7)$$

Dimana $IDCG_p$ merupakan DCG yang dikalkulasikan dengan urutan ideal, dimana item yang paling relevan tersusun pada posisi atas dan p merupakan posisi dimana DCG dikalkulasikan. Sehingga dapat dilakukan observasi bagaimana kemiripan ururan yang diperoleh dengan urutan yang ideal [5].

3. HASIL DAN PEMBAHASAN

Pengujian algoritma dilakukan dengan perhitungan nilai *precision*, *recall*, dan *normalized Discounted Cumulative Gain* (nDCG) yang diambil dari pengujian terhadap nilai jumlah kueri (k). Dataset yang digunakan dalam pengujian merupakan dataset sampel yang telah diproses dengan jumlah isi sebanyak 99.316 baris data *podcast* yang dipublikasi pada bulan Desember tahun 2017 pada *website* www.listennotes.com. Total kategori pada *podcast* adalah 67 kategori. Pengujian akan dilakukan pada sistem dengan menggunakan parameter jumlah kueri (k) yang berbeda-beda. Gambar 1 (a) menunjukkan distribusi nilai presisi dari hasil rekomendasi dengan menggunakan *n-gram* dan *term frequency* (b) menunjukkan distribusi nilai *recall* dari hasil rekomendasi dengan menggunakan *n-gram* dan *term frequency* (c) menunjukkan distribusi nilai f1-*score* dari hasil rekomendasi dengan menggunakan *n-gram* dan *term frequency*.

(a)



(b)

(c)

Gambar 1 (a) Distribusi Presisi (b) Distribusi *Recall* (c) Distribusi F1-*Score*

Gambar 1 menunjukkan dominasi kategori *podcast* memiliki nilai presisi, *recall* dan f1-*score* yang rendah. Hal ini berarti judul dan deskripsi pada *podcast* cukup sulit digunakan sebagai fitur untuk memberikan rekomendasi. Namun ada beberapa kategori yang memiliki nilai yang cukup baik. Tabel 1 menunjukkan rata-rata sepuluh nilai f1-*score* terbaik dari 67 kategori yang diamati.

Tabel 1 Top 10 Rata-rata F1-*score* terbaik berdasarkan Kategori

Nama Kategori	*Precision*	*Recall*	F1-score
Islam	0.288	0.129	0.178
Christianity	0.315	0.113	0.166
Religion & Spirituality	0.304	0.109	0.16
Spirituality	0.304	0.109	0.16
Judaism	0.266	0.106	0.152
Buddhism	0.257	0.091	0.134
Philosophy	0.22	0.078	0.115
Design	0.212	0.079	0.115
Hinduism	0.182	0.081	0.112
Other	0.214	0.075	0.111

Nilai rata-rata *F1-score* dan rata-rata *recall* terbaik dihasilkan oleh kategori *Islam,* rata-rata *Precision* tertinggi dihasilkan oleh kategori *Christianity*. Kesamaan yang terdapat pada kategori-kategori yang mampu menghasilkan nilai tertinggi seperti *Hinduism, Christianity,* dan *Islam* adalah *podcast* pada kategori-kategori tersebut banyak menggunakan istilah spesifik yang hanya dimiliki oleh kategori tersebut pada judul dan deskripsinya. Istilah-istilah spesifik tersebut mampu memberikan rekomendasi yang lebih baik. Rata-rata nilai f1-*score* terbaik dari top 10 kategori *podcast* adalah 14%. Tabel 2 menunjukkan hasil rata-rata *nDCG, Precision, Recall* berdasarkan jumlah kueri dan jumlah rekomendasi.

Tabel 2 Tabel Rata-rata nDCG, *Precision*, *Recall* berdasarkan jumlah kueri dan jumlah rekomendasi

Jumlah kueri	Jumlah Rekomendasi	*nDCG*	*Precision*	*Recall*
3	15	0.449	0.188	0.067
3	30	0.465	0.168	0.111
3	45	0.468	0.159	0.142
3	60	0.469	0.155	0.155
5	15	0.483	0.204	0.048
5	30	0.498	0.181	0.083

Jumlah kueri	Jumlah Rekomendasi	*nDCG*	*Precision*	*Recall*
5	45	0.503	0.169	0.113
5	60	0.505	0.161	0.136
7	15	0.505	0.218	0.038
7	30	0.517	0.192	0.068
7	45	0.522	0.179	0.091
7	60	0.525	0.169	0.113
9	15	0.520	0.229	0.032
9	30	0.530	0.201	0.057
9	45	0.536	0.187	0.076
9	60	0.538	0.177	0.095

Dari tabel 2 dapat disimpulkan bahwa dengan menaikkan jumlah kueri dan jumlah rekomendasi akan dapat meningkatkan nilai *nDCG* yang lebih tinggi yaitu dengan jumlah kueri 9 dan jumlah rekomendasi sebanyak 60 menghasilkan nilai nDCG sebesar 53.8%. Namun dengan nilai meningkatkan jumlah kueri akan dapat mengakibatkan terjadinya *out of bound* bagi *podcast* dengan judul dan deskripsi yang pendek / lebih kecil dari yang diujikan. Pada tabel 3 dapat dilihat jumlah data yang mengalami *out of bound* ketika diuji dengan jumlah kueri tertentu.

Tabel 3 Jumlah *Podcast out of bound* berdasarkan jumlah kueri

Jumlah kueri	Jumlah *Podcast out of bound*
3	4815
5	12456
7	21351
9	29164

4. KESIMPULAN

Berdasarkan hasil implementasi dan pengujian yang dilakukan, maka kesimpulan yang diperoleh adalah:

1. Nilai rata-rata tingkat relevansi rekomendasi *podcast* yang mampu dihasilkan adalah sebesar 53.8%. Nilai relevansi tertinggi ini diperoleh ketika rekomendasi dijalankan dengan pengaturan jumlah kueri yang digunakan sebanyak 9 kueri dan hasil rekomendasi sebanyak 60 *podcast*.
2. Nilai rata-rata f1-score dari sepuluh kategori terbaik di *podcast* adalah 14% dengan f1-*score* terbaik sebesar 17.8% pada kategori Islam.

5. SARAN

Saran untuk membantu penelitian selanjutnya adalah dengan menerapkan *Natural Language Processing* karena metode yang saat ini digunakan belum menerapkan *Natural Language Processing* sehingga sistem hanya mampu bekerja pada konten yang ditulis dengan huruf latin dalam bahasa Inggris. Oleh karena itu dengan menerapkan *Natural Language Processing* pada penelitian selanjutnya, sistem akan dapat diterapkan pada konten-konten yang tertulis dalam bahasa selain Bahasa Inggris ataupun huruf non-Latin.

DAFTAR PUSTAKA

[1] L. Yang, Y. Wang, D. Dunne, M. Sobolev, M. Naaman and D. Estrin, "More Than Just Words: Modeling Non-Textual Characteristics of Podcasts," *WSDM'19 Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining,* pp. 276-284, 2019.

[2] Edison Research and Triton Digital, "The Infinite Dial 2019," 6 March 2019. [Online]. Available: https://www.edisonresearch.com/infinite-dial-2019/. [Accessed 24 March 2019].

[3] M. Locker, "Apple's podcasts just topped 50 billion all-time downloads and streams," 25 April 2018. [Online]. Available: https://www.fastcompany.com/40563318/apples-podcasts-just-topped-50-billion-all-time-downloads-and-streams. [Accessed 24 March 2019].

[4] Z. Xing, M. Parandehgheibi, F. Xiao, N. Kulkarni and C. Poulio, "Content-based Recommendation for Podcast Audio-items using Natural Language Processing Techniques," *2016 IEEE International Conference on Big Data (Big Data),* 2016.

[5] C. Nascimento, A. H. Laender, A. S. da Silva and M. A. Gonçalves, "A Source Independent Framework for Research Paper Recommendation," Ottawa, 2011.

[6] A. G. Jivani, "A Comparative Study of Stemming Algorithms," *International Journal of Computer Technology and Applications,* vol. II, 2011.

[7] M. F. Porter, "An algoritma for suffix stripping," *Readings in information retrieval ,* pp. 313-316, 1997.

[8] W. B. Cavnar and J. M. Trenkle, "N-Gram Based Text Categorization," 2001.

[9] J. Pustejovsky, "Introduction to N-grams Models," 2015.

[10] F. C. Pereira, Y. Singer and N. Tishby, "Beyond Word N-Grams," 1995.

[11] N. Zanini and V. Dhawan, "Text Mining: An introduction to theory and some applications," *Research Matters: A Cambridge Assessment publication,* 2015.

[12] A. Rajaraman and J. D. Ulman, "Data Mining," in *Mining of Massive Datasets*, 2011.

[13] C. Nascimento, A. H. F. Laender and A. S. da Silva, "A Source Independent Framework for Research Paper Recommendation," Ottawa, 2011.

[14] A. Singhal, "Modern Information Retrieval : A Brief Overview," 2001.

[15] D. M. W. Powers, "Evaluation : From Precision, Recall and F-Factor," 2007.

[16] C. D. Manning, P. Raghavan and H. Schütze, "Evaluation in Information Retrieval," in *Introduction to Information Retrieval*, Cambridge University Press, 2008, pp. 139-161.