A DeepQA Based Real-Time Document Recommender System

Joshua E. Dzitiro*, Edgar Jembere† and Anban W. Pillay‡
Centre for Artificial Intelligence Research, University of KwaZulu-Natal
University of KwaZulu-Natal, Durban, 4041, South Africa
[1]joshuadzitiro@gmail.com
[2]ejembere@gmail.com
[3]pillayw4@ukzn.ac.za

***Abstract*—Recommending relevant documents to users in real-time as they compose their own documents differs from the traditional task of recommending products to users. Variation in the users' interests as they work on their documents can undermine the effectiveness of classical recommender system techniques that depend heavily on off-line data. This necessitates the use of real-time data gathered as the user is composing a document to determine which documents the user will most likely be interested in. Classical methodologies for evaluating recommender systems are not appropriate for this problem. This paper proposed a methodology for evaluating real-time document recommender system solutions. The proposed methodology was then used to show that a solution that anticipates a user's interest and makes only high confidence recommendations performs better than a classical content-based filtering solution. The results obtained using the proposed methodology confirmed that there is a need for a new breed of recommender systems algorithms for real-time document recommender systems that can anticipate the user's interest and make only high confidence recommendations.**

***Index Terms*—Real-time Document Recommendation, Content-Based Filtering, Lantent Dirichlet Allocation, Test Collection**

I. Introduction

We envisage a real-time document recommender system that recommends relevant documents to the user in real time as the user is composing a document. The input to the system is the sentences the user is working on and the output is a ranked list of documents that are relevant in that context. This task is different from the traditional task of recommending products to users as variation in the discussion context as the user types a document can undermine the effectiveness of classical recommender system techniques. Traditionally, recommender systems use offline data to infer user preferences. Offline data is not rich enough to capture the discussion context and information needed to anticipate the documents the user may be interested in. A real-time document recommender system needs to capture the user's real-time interest and adapt to them as the user moves from one context to the other.

An important consideration for the system is to be able to return high precision recommendations so as to reduce the information overload on the user. Recommending too many documents, with low relevancy, will be disruptive. Apart from the need to adapt to the user's interests in real-time, this paper argues that it is also important that a real-time recommender system recommends as few items as possible, but ensure that it has high confidence that the user will be interested in the documents it recommends.

A crucial first step in the development of such systems is the establishment of an evaluation protocol. Owing to the novelty of the problem being addressed in this paper, a standard methodology for evaluating such a solution was not available. An *in-situ*, user-oriented evaluation was not feasible for this experimental study. This work develops an *ex-vivo*, system-oriented evaluation methodology suitable for a real-time document recommender system.

The proposed methodology was used to investigate whether a Content-Based Filtering algorithm that anticipates user interests based on relatedness of discussion topics fares better than a classical Content-Based Filtering algorithm in real-time document recommendation. We also used the proposed methodology to investigate whether an Anticipative Content-based Filtering algorithm, which only makes high confidence recommendations, will do better in terms of its precision.

The DeepQA approach [1] was used to develop the algorithm that makes high confidence recommendations. DeepQA is a software architecture developed by IBM for deep content analysis and evidence-based reasoning.

This paper is organised as follows: Section II reviews related work and focuses on techniques that have been adapted in this study. Section III discribes the development of an Anticipative Real-Time Document Recommender System algorithm using the DeepQA approach. Section IV discusses the proposed evaluation methodology and describes how it was used to compare the different recommender system algorithms considered in this study. Section V gives a discussion of the results and future research directions.

II. Related Work

Not much work on real-time document recommender systems has been reported in literature. The only known existing system is Google Explore. However, no publicly available research articles on this system is available. Research on recommender systems that solve a similar problem has emanated from online entertainment stores such as Netflix, that capture the user's real-time interest in order to make recommendations. Examples of such work include [2]–[4]. The focus in

such papers is mainly on how to get contextual information from real-time data streams and using the contextual information to adapt to user's interests. The evaluation methodology used for these systems are, however, not appropriate for our use case.

The evaluation methodology proposed in this paper draws inspiration from the concept of Test Collections used in evaluating Information Retrieval (IR) systems [5], [6]. The main components of a Test Collection are a standardised document collection, topics, and relevance assessments. These, combined with evaluation metrics, simulate the users of an IR system in an operational setting that enables effective evaluation of the system [6]. Test Collections are a very important component of IR research, hence several conferences (e.g. TREC[1], CLEF[2] and NTCIR[3]) dedicated to the creation and use of Test Collections were established. However, for researchers who wish to deploy an IR system in an operational setting where no test collections exists, the IR Systems community has over the years developed a tried and tested methodology for building a Test Collection "from scratch". We adapted this methodology to build a Test Collection for evaluating real-time document recommender system, with a few deviations on how the list of relevant documents is created.

Generally, document recommender systems are based on Content-Based (CB) filtering [7]. CB filtering systems analyze item descriptions to identify items that are of particular interest to the user. They recommend items based on a comparison between the content of the items and a user profile. The content of each item is usually represented as a set of textual features, and for document recommender systems these features are typically the words that occur in the document. Key to CB filtering in these recommender systems is the ability to efficiently extract discriminative textual features to be used by the algorithms. Feature extraction for text processing mainly use the TF-IDF metric [8]. The TF-IDF metric creates a feature space from the words that have high frequency in the document collection but appear in very few documents. Latent Dirichlet Allocation (LDA) [9] has also emerged as a technique that can be used for creating a feature space in text processing. LDA is a generative statistical model that enables a document to be viewed as mixture of a small number of topics that characterise the document. The topics are represented by a set of words that enable documents to be mapped to topics to a certain degree of certainty. We used both the concepts of TF-IDF and LDA to create features spaces for document comparison in formulating the Anticipative Content-Based (A-CB) filtering algorithm. However, the techniques where applied at different stages of the algorithm.

[1]https://trec.nist.gov/
[2]http://www.clef-initiative.eu/
[3]http://research.nii.ac.jp/ntcir/index-en.html

III. Real-Time Document Recommender Systems

A. Design Requirements

A real-time document recommendation algorithm should use information extracted in real-time, as the user is typing, to generate a list of candidate document recommendations. The candidate documents should be similar to what the user has typed. However, we emphasise that in real-time document recommendation, similarity to what has been typed does not imply relevance to the user. Relevance to the user is achieved when the system is able to anticipate what the user is more likely to include in the document and recommend documents that are in the anticipated topics. Thus, the second design requirement for the solution is that it should be able to anticipate the topic that the user is more likely to type, and recommend documents that are in the anticipated topics. To reduce the number of documents recommended to the user and ensure that these documents are relevant, there is a need for the solution to recommend only documents the system is sure are relevant to the user, at least to some degree of confidence.

B. A DeepQA based Solution to Real-time Document Recommendation

The DeepQA approach dictates that a question-answering solution should have three major subsystems (i) Candidate answer/hypothesis generation, (ii) Hypothesis testing and evidence scoring and (iii) Confidence merging and ranking. The approach uses an ensemble of machine learning techniques in each of these subsystems. However, to reduce bias on the candidates generated, the DeepQA approach prescribes that the techniques used for hypothesis generation should not be used for hypothesis testing. This study takes inspiration from this approach to provide a solution for real-time document recommendation.

1) Hypothesis Generation: The purpose of hypothesis generation is to identify the most promising documents that might be of interest to the user.

This done by first creating the *term frequency–inverse document frequency* (TF-IDF) vectors for the documents in the repository and the text that the user has typed so far (query). TF-IDF is a product of two measures: Term Frequency(TF) and Inverse Document Frequency(IDF). TF is the number of times a word w appears in a document d and IDF is calculated by taking the logarithm of the quotient of dividing the total number of documents in the corpus N by the number of documents d_w that have a word w. Thus, the TF-IDF of a word w in document d contained in a corpus of N documents is given by Equation (1):

$$\text{tf}-\text{idf}\,(w,d) = \text{tf}\,(w,d) \times \text{idf}\,(w) = \text{tf}\,(w,d) \times \log\left(\frac{N}{n_w}\right) \quad (1)$$

The TF-IDF vector of a document is a set of the TF-IDFs for all discriminative words in the document. After the TF-IDF vectors of all the documents in the corpus are created, they are used determine how similar each document in the

repository is to the query. This is done using the Pearson Correlation similarity measure given in Equation (2). The Pearson Correlation between a document's vector D and a query's vector Q is given by:

$$\text{corr}_{DQ} = \frac{n\sum D_i Q_j - \sum D_i \sum Q_j}{\sqrt{n\sum {D_i}^2 - \left(\sum D_j\right)^2} * \sqrt{n\sum {Q_i}^2 - \left(\sum Q_j\right)^2}} \quad (2)$$

Once the Pearson Correlation between the documents and the query is calculated, the top 100 documents in terms of their similarity to the query are returned to serve as the candidate hypothesis for recommendation. The algorithm for hypothesis generation is shown in Algorithm (1). The returned documents are then subjected to hypothesis testing.

Algorithm 1: Hypothesis generation

Data: documents, query (user's text)

Result: *topDocuments*

```
begin
    /* Query Vector Generation */
    generate TF-IDF vector for the query
    foreach document in documents do
        document.similarity = calculate the similarity
          between the TF-IDF vectors of the document and
          the query
    rank the documents using document.similarity
    topDocuments = the top n documents
```

2) Hypothesis testing and Evidence Scoring: In hypothesis testing the system searches for more evidence that the user will be interested in the candidate documents. Each document's evidence score is determined by its *relevance* to what the user is currently discussing and what the system anticipates the user will discuss next, which we refer to as *anticipation*.

To determine what a document or a query discusses, we propose using Topic models. Topic models are probabilistic models that uncovers hidden semantic structures(topics) of a corpus and, each document in the corpus is composed of such topics. LDA was used to create the topic models in this research.

LDA assumes that there is a fixed number of *topics*, k, according to which documents are generated. Each topic is characterised by a multinomial distribution over the words in the vocabulary. A document is generated by sampling a mixture of these topics and then sampling words from that mixture. Formally, a document of N words, $w = \langle w_1, w_2, w_3, ...w_N \rangle$ is generated by the following process [9]:

1) First, $\Theta = (\theta_1, \theta_2, \theta_3, ..., \theta_k)$ is sampled from a $Dirichlet(\alpha_1, \alpha_2, \alpha_3, ..., \alpha_k)$ distribution, where $\theta_i \geq 0$, and $\sum_i \theta_i = 1$.
2) for each of the N words, a topic $t_n \in \{1, 2, 3, ..., k\}$ is sampled from a Multinimial didtribution, $p(t_n = i|\Theta) = \theta_i$.
3) Finally each of the word w_n is sampled, conditionally on the t_n^{th} topic, from the multinomial distribution $p(w|t_n)$.

Given hyper-parameters, α and β, LDA defines a joint probability distribution of the topic mixture Θ, a set of N topics **t** and a document **w** as shown in Equation (3).

$$p(\Theta, \mathbf{t}, \mathbf{w}|\alpha, \beta) = p(\Theta|\alpha) \prod_{n=1}^{N} p(t_n|\Theta) p(w_n|t_n, \beta) \quad (3)$$

The marginal probability distribution of a document **w** is found by summing over **t** and integrating over Θ to give Equation (4).

$$p(\mathbf{w}|\alpha, \beta) = \int p\left(\Theta|\alpha\right) \left(\prod_{n=1}^{N} \sum_{t_n} p\left(t_n|\Theta\right) p\left(w_n|t_n, \beta\right) \right) d\Theta \quad (4)$$

The hyper-parameters α and β are corpus level parameters, assumed to be sampled once in the process of generating a corpus. Equation (4) gives the Topic Distribution Vector (TDV). The TDV is a vector composed of a set of probabilities showing how likely is a given document to belong to each of the k topics.

For determining *relevance*, the idea that what the user is going to type next, is most likely not very different from what the user has typed this far, was used. Thus, the relevance of a document to a query is determined by its similarity to the query's text. A document's relevance score was therefore computed by measuring the similarity between the document's TDV and the query's TDV.

For determining *anticipation*, we use the idea that, as the user moves from one paragraph to another, the user will be moving from one topic to another related topic. We postulate that the topic with the highest probability in the TDV for the query passage, is the one that the user is currently discussing, and the topic the user is likely to discuss next is related the current topic. We took advantage of this idea to create an Adjusted Topic Distribution Vector(ATDV), which we used to represent the system's anticipation of what the user will discuss next. The ATDV was created as follows:

1) First, the topic with the highest probability (THP) in the TDV, and its Most Related Topic are identified. The most related topic is identified by taking all the words that belong to THP and use them to get THP's TDV from the Topic Model. The topic with the second highest probability in the THP's TDV corresponds to the Most Related Topic. The THP will have the highest probability in it's TDV.
2) The probability of THP in the query's TDV is reduced by moving some proportion of it's probability mass to the Most Related Topic. The resulting vector is the ATDV for the query.

The anticipation value for each document is calculated using the Pearson correlation similarity between the document's TDV and the ATDV.

The complete algorithm for Hypothesis testing and Evidence scoring is shown in Algorithm (2). The relevance and the anticipation scores from Algorithm (2) will be used in the Confidence merging and ranking component.

Algorithm 2: Hypothesis Testing and Evidence Scoring

```
Data: documents, query (user's text)
Result: documents

foreach document in topdocs do
    generate the TDV for the document;
end

foreach document in topdocs do
    document.relevance = calculate the similarity between
      the TDV of the document and the query;
end

foreach document in topdocs do
    document.anticipation = calculate the similarity between
      the TM vectors of the document and the ATDV of the
      query;
end
```

3) Confidence Merging and Ranking:: The system uses confidence merging and ranking to ensure that the documents that are recommended are relevant to what the user is discussing (relevance), and also relevant to what the system anticipates the user will discuss next (anticipation). The system ranks documents by their anticipation values and recommends documents that have a relevance value that is above a given relevance threshold. The algorithm for confidence merging and ranking is given in Algorithm (3).

Algorithm 3: Confidence Merging and Ranking

```
Data: topDocuments, query (user's text)
Result: recommendations

rank the topdocs using document.anticipation;

list recommendations;

foreach document in topdocs and rank < 10 do
    if document.relevance > minRelevance then
        add document to recommendations
    end
end

return recommendations
```

IV. Evaluation

Evaluation of the system will seek to test the following hypotheses related to Anticipative Real-Time Document recommender algorithms:

1) Anticipative Content-based filtering algorithms have the potential of making better recommendations compared to the classical Content-Based filtering algorithm,
2) Enhancing Anticipative CB filtering algorithms with confidence improves their precision, and
3) An Anticipative Content-based algorithm is scalable with the amount of text in the query passage.

A. Experimental Platform

Evaluation of a real-time document recommender system is not straight forward due to it being open domain problem. Researchers often resort to online human evaluation of the responses given by a system in open domains. However, such evaluations are typically time consuming and labour intensive. This research takes inspiration from evaluation of information retrieval systems in controlled laboratory-based settings. Unfortunately, no benchmark Test Collections exist for our use case. Thus, a Test Collection for evaluating real-time document recommender systems was created. To simulate an operational environment for the system we propose that the Test Collection should have the following two components; (i) the document collection, (ii) a set of test documents with their corresponding query passages, Query relevant (QREL) passages, and QREL lists. A QREL passage is a passage extracted from a given test document, whose related documents should be recommended by the real-time document recommender algorithm in response to a given query passage. A QREL list is a list of documents that are deemed relevant to the query passage based on their topical relatedness to the QREL passage. Thus, the QREL list creates the ground truth for evaluation of the algorithms.

1) Document Collections: The document collection was created using 1000 randomly selected open access Computer Science(CS) papers published by Elsevier, between 2014 and 2017. Elsevier classifies the following areas within CS: Artificial Intelligence, Computational Theory and Mathematics, Computer Graphics and Computer-Aided Design, Computer Networks and Communications, Computer Vision and Pattern Recognition, Computer Hardware and Architecture, Human-Computer Interaction, Information Systems, Signal Processing, and Software. This wide range of topics makes the test collection wide enough to simulate an operational environment.

2) Test Documents and Query Formulation: The queries were created by taking the first four paragraphs from 8 randomly selected CS papers published by Elsevier in 2018. The first 4 paragraphs were used for creating the query passages.

3) QREL Passage and QREL List: The relevance assessment was done using the QREL passage as the source of the ground truth. The ground truth was created using documents that are semantically similar to the 5th paragraphs of a given Test Document. The 5th paragraphs represent what the user is going to type next. The QREL list for a given Test Document was created by calculating the Pearson Correlation between its QREL passage's TDV and the TDV of each of the documents in the Document Repository. The documents with the correlation above 0.7 were then placed in the QREL list. Figure (1) shows how the QREL list was created,

B. Experimental Results

All algorithms evaluated in this paper were implemented in Java. The MAChine Learning for LanguagE Toolkit (MAL-

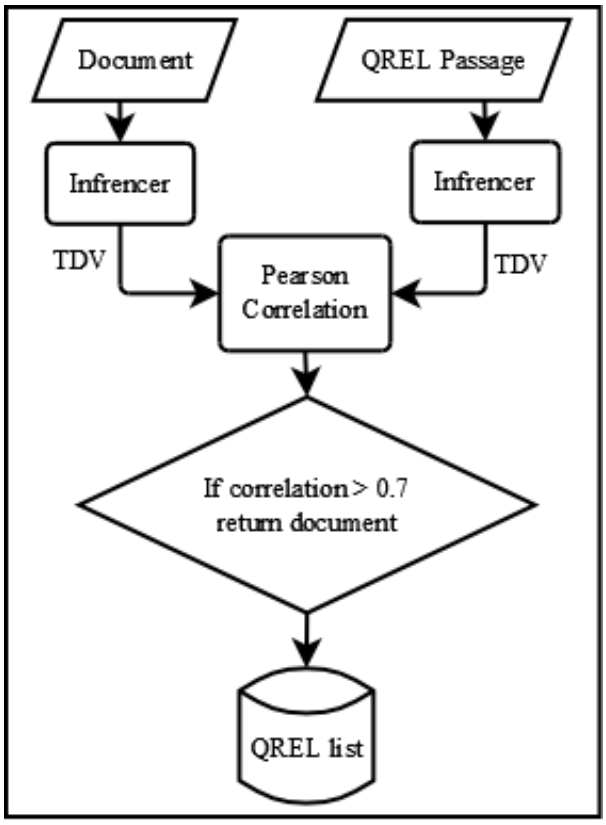

Fig. 1. Algorithm for creating the QREL List

LET)[4], was used for implementation of LDA. All experiments were conducted on an Intel i7,Quant Core, 3GHz, processor, with 8 GB memory.

The first set of experiments conducted were for determining the best configuration of our experimental platform. Different query lengths were used, starting from 1 to 5 paragraphs, to form the query passage. This was done for each of the 8 test documents and it was discovered that using paragraphs 1-4 as the query text and paragraph 5 for creating the QREL list yielded the best results in terms of precision for both the classical CB filtering and the A-CB filtering algorithms. We also ran experiments to determine the best factor for redistributing the probability mass from query topic to the most related topic. Our experiments showed that for the above-mentioned query length, the best factor was 0.3. This means that the A-CB filtering algorithm was taking 0.3 of the probability mass allocated by the topic model to the topic which was most likely to be the correct topic of the query, and allocates the probability mass to its most related topic.

1) Recall-Precision(RP) Curve- Area under the RP curve: The afore-discussed experimental configuration was used for this experiment. For each of the 8 documents in the test set 10 runs were done and the average recall and precision for each rank were recorded.

Figure 2 shows the results of a comparative analysis of the classical CB filtering algorithm to the A-CB filtering algorithm. The number of retrievals was restricted to twenty

[4]http://mallet.cs.umass.edu/

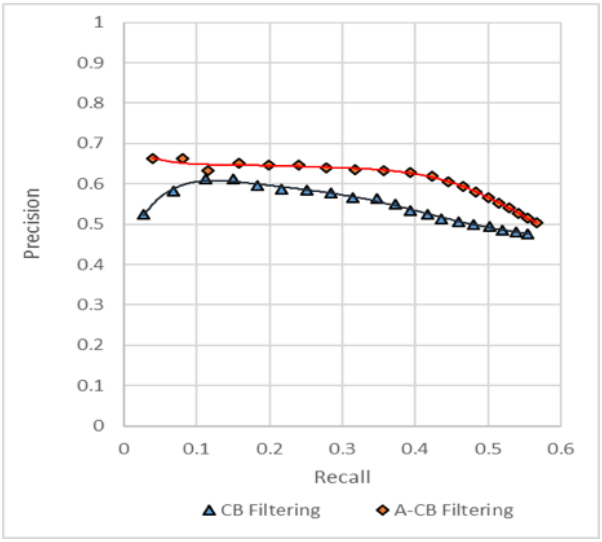

Fig. 2. RP Curve for Comparing the CB filtering and the A-CB filtering Algorithms

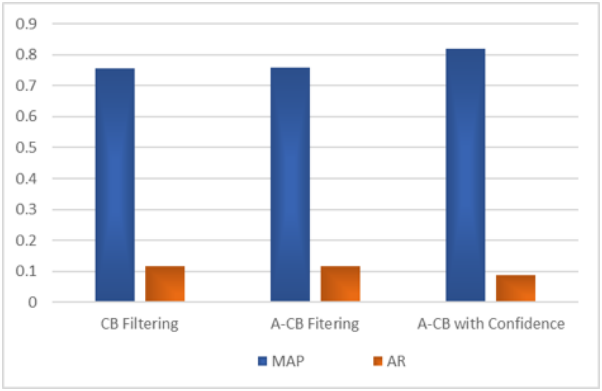

Fig. 3. Comparison of the CB filtering, A-CB filtering and the CA-CB filtering Algorithms in terms of MAP

due to the large number of relevant documents in the test collection. This resulted in a low recall. For Test Collection based evaluations, precision is considered to be more informative metric than recall. The results show that the A-CB filtering algorithm has more area under the RP curve. This implies that it performs better than the CB filtering algorithm. This establishes the need for the development of anticipative recommender systems algorithms for real-time document recommender systems.

Figure 3 shows the comparison of the classical CB filtering, the A-CB filtering, and the Anticipative Content-based filtering with Confidence (CA-CB filtering) in terms of the Average Recall (AR) and Mean Average Precision (MAP). The CA-CB filtering algorithm only makes high confidence recommendations. The results show that the CA-CB filtering recommendation tend to have a higher MAP than the other two approaches. These results show that recommending only the documents that the system has some level of confidence that the user will be interested in, has the potential of improving the precision of a real-time document recommender system. The recall for the CA-CB filtering was the lowest.

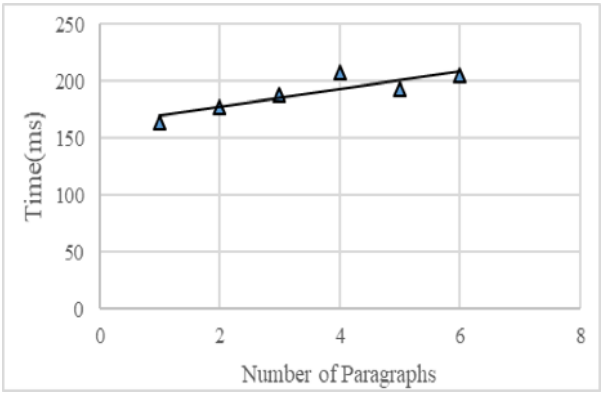

Fig. 4. Scalability of the A-CB with Increases in the Query Passage Size

This is expected since the algorithm returns the least number of document. Higher precision for CA-CB algorithm implies the algorithm is doing better than the other algorithms in making sure that there are fewer irrelevant document recommended to the user.

2) Scalability of the proposed Solution with increases in the query text: Figure 4 shows the results of the scalability analysis of an A-CB filtering algorithm with increases in the query size. The results show that the algorithm has linear time complexity with increases in the query size.

V. Conclusions and Future Work

This paper aimed to develop a methodology for evaluating real-time document recommender systems and subsequently use the methodology to establish the need for a new breed of real-time document recommendation algorithms with the following qualities; (i) ability to anticipate user changes in user interest in Real-Time as the user works on a document, (ii) ability to only make high confidence recommendations to avoid overloading the user. In order to anticipate user interest, LDA was used to anticipate discussion topics. The DeepQA approach was used to enable high confidence recommendations. The proposed document recommender system algorithm was in its most basic and simplest form, but it was still found to perform better than the classical content-based filtering algorithm. This confirmed our hypothesis that variation in the user's interests as the user works on the document can undermine the effectiveness of classical content-based filtering algorithms in Real-Time document recommender systems. The results also showed that even though an anticipative content-based filtering algorithm may not be able to recommend only the relevant documents, it tends to recommend relevant documents higher up the recommendation list. This is a very important result, which suggest that filtering out low confidence recommendation does not compromise the overall quality of an anticipative content-based filtering algorithm. In fact, the results also show that filtering out low confidence recommendations increases precision at a very low cost on recall.

As part of our future work we will seek to improve the creation of the QREL list by using citations in the QREL passage to create the QREL list. This is intuitively a better way of creating the ground truth for evaluation of Real-Time Document Recommender systems. One of the major results from this study is the finding that there is a need for a new breed of Content-Based filtering algorithms with the ability to anticipate user interest based on contextual session data. We will in future investigate how Sequential Topic modelling and techniques used in Topic Drift Detection can be used to develop of Anticipative Content-Based filtering algorithms for Real-Time Document Recommender Systems.

References

[1] D. Ferrucci, E. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. A. Kalyanpur, A. Lally, J. W. Murdock, E. Nyberg, J. Prager, N. Schlaefer, and C. Welty, "Building watson: An overview of the DeepQA project," *AI Magazine*, vol. 31, no. 3, 2010.

[2] W. Yuan-hong and T. Xiao-qiu, "A real-time recommender system based on hybrid collaborative filtering," in *ICCSE 2010 - 5th International Conference on Computer Science and Education, Final Program and Book of Abstracts*, 08 2010.

[3] E. Lacic, "Real-time recommendations in a multi-domain environment," in *Extended Proceedings of the 27th ACM Conference on Hypertext and Social Media (HT'2016)*. ACM, 2016.

[4] X. Amatriain, "Big & personal: Data and models behind netflix recommendations," in *Proceedings of the 2nd International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications*, ser. BigMine '13. New York, NY, USA: ACM, 2013, pp. 1–6. [Online]. Available: http://doi.acm.org/10.1145/2501221.2501222

[5] M. Sanderson and M. Braschler, "Best practices for test collection creation and information retrieval system evaluation," TrebleCLEF, Tech. Rep., 2009.

[6] P. Clough and M. Sanderson, "Evaluating the performance of information retrieval systems using test collections," *Information Research*, vol. 18, no. 2, June 2013. [Online]. Available: http://eprints.whiterose.ac.uk/78886/

[7] M. J. Pazzani and D. Billsus, "Content-based recommendation systems," in *The Adaptive Web*, P. Brusilovsky, A. Kobsa, and W. Nejdl, Eds. Berlin, Heidelberg: Springer-Verlag, 2007, ch. Content-based Recommendation Systems, pp. 325–341. [Online]. Available: http://dl.acm.org/citation.cfm?id=1768197.1768209

[8] H. C. Wu, R. W. P. Luk, K. F. Wong, and K. L. Kwok, "Interpreting tf-idf term weights as making relevance decisions," *ACM Trans. Inf. Syst.*, vol. 26, no. 3, pp. 13:1–13:37, Jun. 2008. [Online]. Available: http://doi.acm.org/10.1145/1361684.1361686

[9] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of Machine Learning Research*, vol. 3, pp. 993–1022, 2003.

Joshua E. Dzitiro holds a BSc in Computer Science and Information Technology and a BSc Honours in Computer Science from University of KwaZulu-Natal. He is currently doing his MSc in Computer Science at the University of KwaZulu-Natal.