Science Concierge: A fast content-based recommendation system for scientific publications

Titipat Achakulvisut*[1], Daniel E. Acuna†[3], Tulakan Ruangrong[4], and Konrad Kording[1,2]

[1]Department of Biomedical Engineering, Northwestern University
[2]Sensory Motor Performance Program, Rehabilitation Institute of Chicago
[3]School of Information Studies, Syracuse University
[4]Department of Biomedical Engineering, Mahidol University

April 4, 2016

1 Abstract

Finding relevant publications or presentations matters to scientists who have to cope with exponentially increasing numbers of published papers. Algorithms help with this task, analogous to the ones used for music, movie, or product recommendations. However, we know little about the performance of these algorithms with scholarly material. Here, we develop an algorithm, and an accompanying Python library, that implements a recommendation system based on the content of voted relevant and irrelevant articles. The algorithm has as design principles the ability to rapidly adapt to new content, provide near-real time suggestions, and be open source. We tested the library on 15K posters from the Society of Neuroscience Conference 2015. We viewed human curated topics assignments as inducing a noisy similarity metric and optimized our algorithm to produce a similarity metric that maximally correlated with the human one. We show that our algorithm significantly outperformed suggestions based on keywords. The work presented here promises to make the exploration of scholarly material faster and more accurate.

2 Introduction

Recommendation systems are routinely used to suggest items based on customers' past preferences. These systems have proven useful for music, movies, news, and retail in general [14]. In contrast, to find new scientific literature, researchers rely mostly on author-provided keywords, titles, author names, and references. These procedures are likely to be biased [6] and also provide a rich get richer (or Matthew) effects [17]. Moreover, this problem is more pronounced during conferences where appropriate keywords may not even exist, let alone citations. An application of recommendation systems to

*titipat.a@u.northwestern.edu

†This work was done while the author was at the Rehabilitation Institute of Chicago and Northwestern University

suggest scholarly material based on the researcher's preferences thus promises to speed up literature search and increase relevance.

There are multiple recommendation systems that use either the personal preferences of a new user (e.g., content-based recommendations) or exploit the similarity between the new users' preferences and previous users' preferences (e.g., collaborative filtering). Most such system are available for commercial software, such as news [13], movie [3], and music [2] applications. In contrast, few projects address scientific literature search. In [23], the authors presented a content-based recommendation system that works on PubMed datasets. In [8], the Scienstein system combined a large set of criteria for providing literature recommendation. In [22], the authors presented a topic-based recommendation system based on a Latent Dirichlet Allocation (LDA) model. It is unclear, however, how well these systems work, and how well they scale to large problems, and their source code is generally not available.

Here we introduce Science Concierge (`github.com/titipata/science_concierge`), an open source Python library that implements a fast and accurate recommendation system for literature search. Briefly, the library uses a scalable vectorization of documents through online Latent Semantic Analysis (LSA) [14]. For the recommendation part, it pairs the Rocchio Algorithm [19] with a large-scale approximate nearest neighbor search based on ball trees [20]. The library aims at providing responsive content-based recommendations utilizing only user's votes. Science Concierge, then, provides an open source solution to content-based scientific discovery.

We tuned and tested the algorithm on a collection of scientific posters from the largest Neuroscience conference in the world, Society for Neuroscience (SfN) 2015. First, we cross-validated the LSA model to capture most of the variance contained in the topics. Second, we tuned the parameters of the algorithm to recommend posters that maximally resembled human curated classifications into poster sessions. We showed that our algorithm significantly outperformed a popular alternative based on keywords, improving suggestions further as it learned more from the user. A front-end interface that uses the algorithm in the back end is available at `http://www.scholarfy.net`, where we used data from Society for Neuroscience (SfN) 2015 conference.

3 Materials and methods

3.1 Conference dataset

Conferences are events that need a rapid understanding of trends. The time between submissions, acceptances, and presentations is typically much shorter than in journals. This makes it crucial for recommendation systems to quickly scan the documents of the conference and let scientists discover materials fast. This is one reason why we focus on testing our system using conference data.

We obtained a license from the Society for Neuroscience (SfN) for the Neuroscience 2015 conference. This is the largest conference in Neuroscience in the world. This dataset included 14,718 posters and talks distributed over 500 sessions spanning 5 days. Not all the content of the conference had abstracts available (e.g., talks) and therefore they were dropped from the analysis. The dataset is not publicly available but an academic license may be requested from the Society (`http://www.sfn.org`).

3.2 Content-based recommendation of scientific documents

There are three main design principles behind Science Concierge. First, it aims at using the content of the documents rather than collaborative filter to prevent the Mathew effect, commonly found in recommendation systems [17]. This effect can be detrimental

for scientific exploration. Second, it aims at proving suggestions as fast as possible. This means that users should get feedback as soon as they vote one item as relevant or irrelevant. Finally, it aims at being validated using some external input. Below we describe the methods to achieve these three goals.

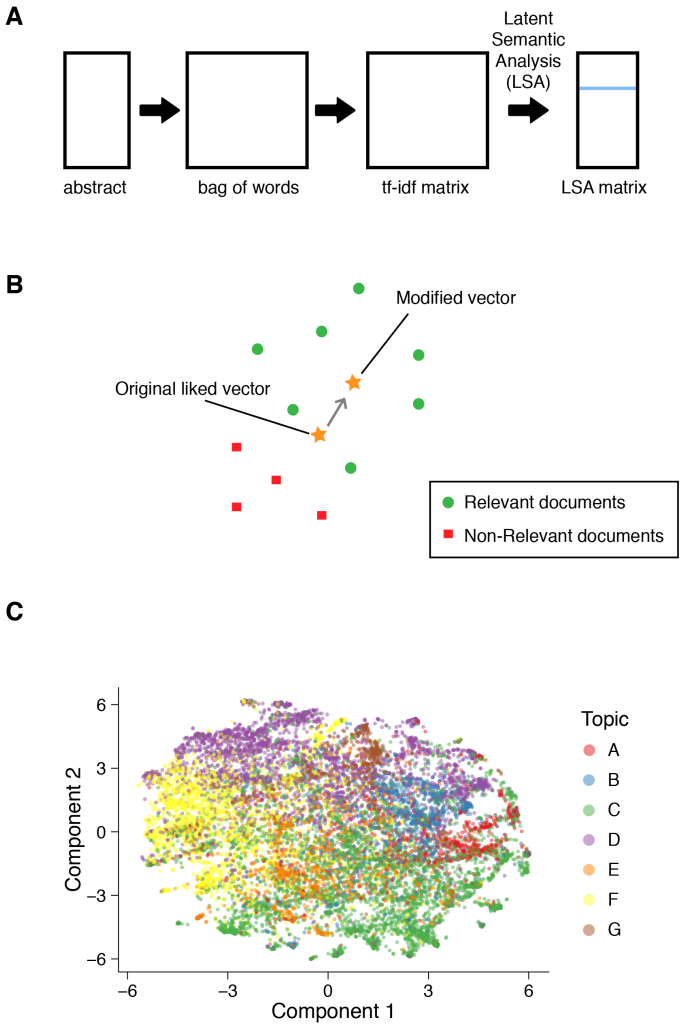

Fig 1. Vector representation of documents (A) Schematic of the workflow for converting abstracts into vector representations (B) Schematic of Rocchio Algorithm (C) Visualization of topics colored by human curated sessions using t-SNE on abstract vectors.

3.3 Text preprocessing

Documents are tokenized by removing English stop words and stemming using the Porter Stemming algorithm [18]. The terms in the documents are uni-grams and bi-grams. This term document matrix is transformed by a term-frequency inverse document-frequency (tf-idf) function. Terms that appear fewer than 3 times were removed, and terms that have tf-idf greater than 0.8 were removed. The intuition behind this way of preprocessing is that we want to focus on words that are indicative of the document that they appear

in, or, said differently, on words that "carry" the topic of the document.

3.4 Latent Semantic Analysis

LSA is a simple topic modeling technique based on singular value decomposition [11]. The tf-idf matrix is transformed using Latent Semantic Analysis (LSA) to reduce noise and improve smoothness. This technique decomposes the document term matrix or, in our case, the sparse tf-idf matrix X as a product of a left singular vector (U), a diagonal singular value matrix (S) and a right singular vector (V) as $X = USV^\top$, where the diagonal singular values in S are ranked from highest to lowest. The number of vectors to choose depends on how accurately we want to capture the matrix X [4]. In our case, this number will be chosen by cross validation. The intuition behind the use of LSA is that we want to discover groups of words that are equivalent in their meaning, as they should load onto the same singular vectors.

The pre-processing is relatively fast on purpose because the algorithm needs to rapidly adapt to changes in the conference. Even with the simple level of preprocessing provided by LSA, we can already start understanding how posters are separated into distinct topical areas. To visualize this, we transform the LSA vectors using a two dimensional reduction by t-Distributed Stochastic Neighbor Embedding (t-SNE) ([21], Fig. 1B), coloring each poster with its curated topical area from letter A through G. From the visualization, it is apparent that some topics are more clearly separable than others, at least as long as we are in a very low dimensional space.

3.5 Poster representation based on keywords

The dataset used here also contained keywords. Searching by keywords is a popular way to discover topics at conferences. We compare our methods to this type of search. In this algorithm, the LSA analysis is applied to the keyword vector of each poster and all the rest of the analysis described below is the same.

3.6 Rocchio Algorithm and Nearest Neighbor Assignment

The Rocchio algorithm is used to produce recommendations based on relevant and non-relevant documents previously voted by the user [19]. The original method works by using the term count vector but instead we use the LSA vectors. The method computes a preference vector by combining the vectors of the voted documents. Finally, the nearest neighbors of such resulting preference vector would be the suggestions.

Mathematically, given a set of documents vectors that are found to be relevant by the user $\{r_1, r_2, \ldots, r_n\}$ and another set of non-relevant documents $\{u_1, u_2, \ldots, u_m\}$, the Rocchio algorithm finds a document vector that combines both types of documents as follows

$$q = q_0 + \frac{\alpha}{N}\sum_{i=1}^{N} r_i - \frac{\beta}{M}\sum_{j=1}^{M} u_j \tag{1}$$

where q_0 was originaly proposed as the mean document vector of an initial query made into the system, such as a search for a particular keyword. The parameters α and β control how much the relevant and non-relevant documents affect the final query vector, respectively. In the case of our system, there is no such initial query q_0 and therefore the mean article replaces this term. The parameters α and β will be found by cross validation using some external validation process.

Once the query vector q has been defined for a user, a nearest neighbor search will be performed to retrieve suggested elements. In our implementation, we use ball trees

to approximate a nearest neighbor search, which tradeoffs speed and accuracy relatively well [20].

3.7 Measuring suggestion performance by using human curated classification

Each poster contained in this dataset has been manually classified into a topic by the organizers of the conference. The topic classification is based on a tree with three levels. For example, a poster might be classified with topic $F.01.r$, where F is the broad area of study in Neuroscience, 01 is a subarea of F, and r is a further subdivision within 01. To validate many of the parameters in our performance tests, we use this topic classification as an indirect measure of how correct the suggestions are.

We implicitly assume that a random attendee would be highly interested in one topic only and not interested in others. This means that users would tend to like poster from the same area. While this assumption may not be accurate for a large number of attendees, we believe in captures the intention behind the classification of topics in this particular SfN conference. To measure the quality of suggestions, then, we ask the system to suggest ten posters based on a set of "liked" posters from a particular topic. For each of the suggestions, we compute the distance between its topic and the topic of the liked posters, using as distance measure the lowest common ancestor in the topic tree [1]. Minimizing the average of the distances between the liked topic and the ten suggested posters will be set as the performance metric for our comparisons.

4 Results

4.1 Parameter optimization

Number of components for LSA

One parameter of the LSA algorithm is the number of components of the SVD decomposition [4], which we find by cross validating on the distance to human curated topic classification as described in Methods. We randomly sampled one poster and liked it. Then, we ask the system to suggest ten posters and compute the average distance in human curated topic of those ten posters to the liked poster (Fig. 2). In this way, we were able to find that the appropriate number of components to be used should be around 100.

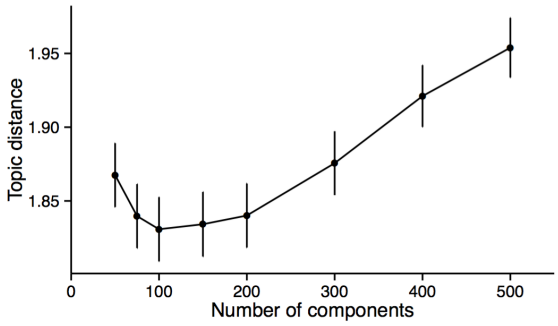

Fig 2. Number of SVD components vs. performance of the algorithm to capture human curated topics

Weight of relevant and non-relevant documents for Rocchio algorithm

The Rocchio algorithm differently weighs the importance of relevant documents with a parameter α and the importance of non-relevant with a parameter β (see Eq. 1). We tuned the parameters α and β using a procedure similar to that used for finding the number of components in SVD.

The experiment this time is done by liking one poster from a random topic and voting as non-relevant a poster from a different topic. We performed three experiments where we tested three distances of the non-relevant voted posters: distance 1 (1 subdivision away), distance 2 (in a difference subarea), and distance 3 (in a different area of study). For each of these experiments, we tried a grid of α and β parameters and computed the average topic distance over a set of 1,000 simulations (Fig. 3). This is possible because we only have two free parameters here.

We found that voting as non-relevant those posters that are 1 distance away produces large effects on performance. Interestingly, we found that the parameter β almost always makes the performance decrease. This means that non-relevant posters should be used to filter out the recommendations rather modifying a user's preference vector. We also found that the best values for the parameter α are always larger than 1. Additionally, we found that non-relevant posters that are 3 distance away in topic space have little effect on performance (Fig. 3). This is intuitive because disliking posters that are far away gives little information about the topic that a user may like. At the end, the best combination of parameters for non-relevant posters that are away 1 distance is $\alpha = 1.8$ and $\beta = 0$.

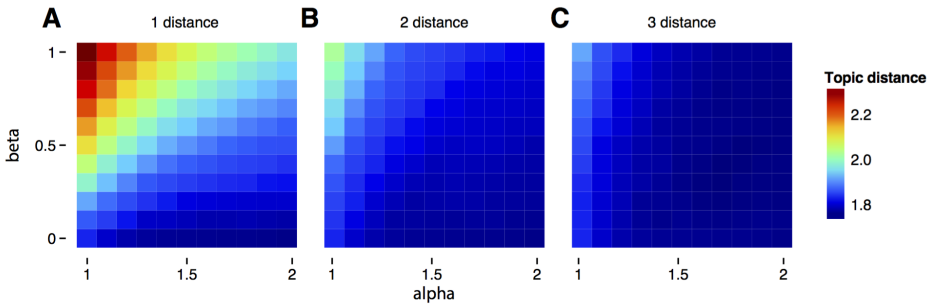

Fig 3. Finding best parameters to weigh relevant and non-relevant votes. Performance of the system as a function alpha and beta parameters for non-relevant documents that are 1 distance away (A), 2 dislike distance away (B), and 3 dislike distance away (C) in human curated topics.

4.2 Algorithm comparison

In this section, we compare the performance of our algorithm against common alternatives. Our algorithm requires the use of abstracts, a significantly more complex data source than keywords, for example. However, using the full abstract could capture variability that is not available in other simpler methods. To properly test the advantages of our approach, we cross validate the performance of our method by using human curated topic distances as a performance metric (see section 3).

For each of the algorithms tested below, we do the following simulation to estimate their performances. For each run of a simulation, we pick a poster at random and vote it as relevant. Then, we ask the algorithm to suggest ten posters based on that vote. We compute the average distance in human curated topic space between the suggestions

and the liked poster. We then vote for another poster randomly selected from the same human curated topic as the first poster. We again ask the algorithm to suggest a set of ten posters and again we compute the average distance. We repeat this process many times to obtain the average distance to human curated topics as a function of the number of votes. This simulation will help us understand the performance of the algorithms as they gather more votes from a simulated user.

As a baseline for all comparisons, we compute the performance of a null model that suggests posters at random. This is necessary because the distribution of human curated topics is not uniform and therefore the distances could be distributed in unexpected ways. Not surprisingly, the average distance to the human curated topic remained constant with the number of votes (Fig. 4, Random) but it was below 3, which is the farthest possible distance. This baseline will allow us to compare performance against a proper null model.

Keywords are a common recommendation technique that relies on using human curated "tags" for each document. In our dataset (see section 3), the authors themselves assigned multiple keywords by picking them from a predefined set crafted by the organizers. We used these keywords to produce recommendations by simply using them as documents. We then model keyword counts by transforming them from the tf-idf matrix to 30 SVD dimensions. Preliminary analysis revealed that 30 dimensions was an appropriate number to capture most of the variability. Suggestions using keywords perform significantly better than random suggestions (paired $t(4999) = 74.74$, $p < 0.0001$) and keyword suggestion performance improves with the number of votes (paired $t(4998) = -4.428$, $p < 0.0001$, Fig. 4). While keywords allow for better suggestions than the null model, authors would need to manually provide them.

Science Concierge uses the abstracts to automatically extract topics and provide suggestions. We found that Science Concierge produces significantly better suggestions than keywords (Fig. 4, paired $t(4999) = 78.95$, $p < 0.0001$). We found that the performance significantly improves as the system learns more about a user ($t(4998) = -31.6, p < 0.001$). Moreover, we found that Science Concierge improves with votes significantly faster than the keyword model ($t(9997) = -66.28$, $p < 0.0001$). Scholar Concierge thus does not require humans to provide keywords and improves the recommendation as more votes are provided.

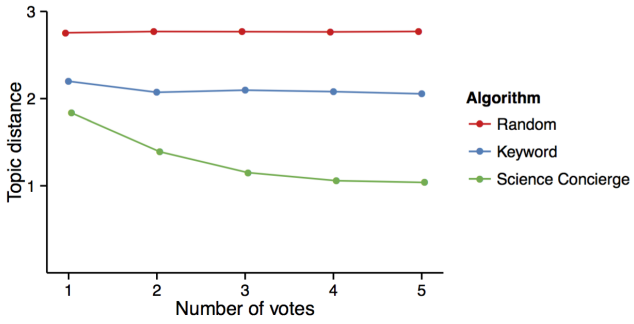

Fig 4. Comparison of algorithms as they learn more from a simulated user. Random suggestions' performance is not necessarily 3 distance away in the topic tree, but remain constant with more votes. Keywords improve suggestions but Science concierge significantly improves upon that and event further with votes.

Ideally, we want distances between documents induced by an algorithm to closely match the distances induced by human curated topics. Here, we tested this idea using the keyword model and the Science Concierge model. This is, we tested the correlation

between the distance of two random documents using a model and the distance of those same documents in human curated topic space. We found that the Spearman rank correlation between these measures was significantly positive for both models ($p < 0.0001$) and that the correlation of the Science Concierge model ($\rho = 0.442$) was significantly higher than the correlation of the keyword model ($\rho = 0.164$), $p < 0.001$. To compare these relationships on a similar scale, we visualized the human curated distance vs. the z-score of the model's distances (Fig. 5). Both measures reproduce human distances, but Science Concierge is better overall.

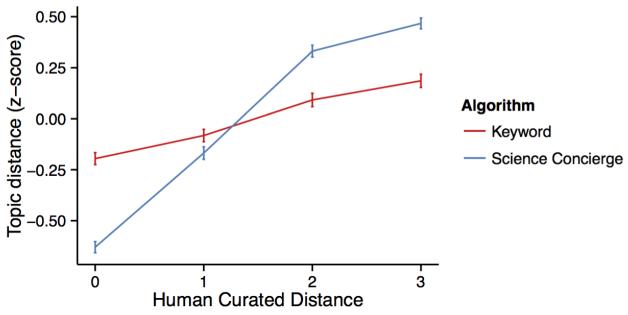

Fig 5. Relationship between human curated distance and topic distance induced by the keyword and Science Concierge models. Both models correlate well, but Science Concierge is significantly better.

5 Discussion and conclusion

Discovering new and relevant scholarly material progressively requires algorithms. For some time now, internet users have enjoyed websites that recommend movies, news, and music. However, the same cannot be said about scientists. They commonly use legacy search systems that cannot learn from previous searchers. In this article, we propose a system that can improve recommendations based on a scientist votes for relevant and irrelevant documents. We tested the system on a set of posters presented at the Society for Neuroscience 2015 conference. We found that our system significantly improves on suggestions based on author-assigned keywords. We also found that our system significantly improves its performance as the user provides more votes. The system returns a complete schedule of posters to visit within 100 ms for a conference of around 15K posters. We publish the source code of the system so others can expand its functionality (`github.com/titipata/science_concierge`). In sum, this article presents a system that can make scientific discovery faster and it is openly available for other scientists to expand.

One surprising finding in our analysis is that the posters voted as non-relevant were better left not influencing the final preference vector. In particular, when voted non-relevant posters were close in topic space to the liked posters, then the performance degraded. If those non-relevant posters were far away, then the performance remains unchanged. In the future, we want to expand the experiment to domains in which a large number of votes is casted. This may allow the algorithm to better understand topic preferences and therefore offer suggestions that exploit the knowledge built in non-relevant votes.

The topic modeling technique used in our algorithm assumes that topics live on a linear space. While this assumption provides significant computational advantages,

there might be cases where more complex modeling approaches may capture more subtle topical relationships. In the future, we will try non-linear probabilistic approaches such as Latent Dirichlet Allocation (LDA) or similar methods [5, 10], which recently have been shown to scale well [9]. However, in our own unreported experiments with this approach we found LDA to be inferior to LSA. To better capture entities embedded in the text, future research will also investigate how to use deep learning modeling of words and phrases [15, 16]. However, it is unclear how these methods cope with scalability. Our system may already provide an appropriate level of speed and accuracy for our domain.

Future research could expand our system in many ways. The Rocchio algorithm's dependency on nearest neighbors makes it inappropriate to exploit the potential richness available on large number of votes [7]. With long term users who provide hundreds or even thousands of votes, it may be more accurate to cast the recommendations as a classification problem [12]. It is unclear however when would be the right time to make such as switch.

Our system proposes a new way to discover scholarly material. Many similar systems (e.g., Google Scholar) do not release their algorithms, making them difficult to study. By opening our algorithm, we will engage the scientific community to collaborate. Moreover, our algorithm does not necessarily need to be constrained to scientific text as any document that can be represented as a vector can be fed into it (e.g., scientific figures, audio). In sum, our system provides an open and fast approach to accurately discover new research.

Acknowledgements

Titipat Achakulvisut was supported by the Royal Thai Government Scholarship grant #50AC002. Daniel E. Acuna was supported by the John Templeton Foundation grant to the Metaknowledge Research Network. We thank SfN for providing the abstract dataset for our tests.

Author Contributions

Titipat Achakulvisut (writing, programming, concept), Daniel E. Acuna (writing, programming, concept), Tulakan Ruangrong (programming), Konrad Kording (writing, concept).

References

[1] Alfred V Aho, John E Hopcroft, and Jeffrey D Ullman. On finding lowest common ancestors in trees. *SIAM Journal on Computing*, 5(1):115–132, 1976.

[2] Muqeet Ali, Christopher C. Johnson, and Alex K. Tang. Parallel collaborative filtering for streaming data, 2011.

[3] Robert M Bell and Yehuda Koren. Lessons from the netflix prize challenge. *ACM SIGKDD Explorations Newsletter*, 9(2):75–79, 2007.

[4] Christopher M Bishop. Pattern recognition and machine learning, 2006.

[5] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *The Journal of Machine Learning Research*, 3:993–1022, 2003.

[6] David Chavalarias and John PA Ioannidis. Science mapping analysis characterizes 235 biases in biomedical research. *Journal of Clinical Epidemiology*, 63(11):1205–1215, 2010.

[7] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics Springer, Berlin, 2001.

[8] Bela Gipp, Jöran Beel, and Christian Hentschel. Scienstein: A research paper recommender system. In *Proceedings of the International Conference on Emerging Trends in Computing (ICETiC'09)*, pages 309–315, 2009.

[9] Matthew Hoffman, Francis R Bach, and David M Blei. Online learning for latent dirichlet allocation. In *Advances in Neural Information Processing Systems*, pages 856–864, 2010.

[10] Andrea Lancichinetti, M Irmak Sirer, Jane X Wang, Daniel Acuna, Konrad Körding, and Luís A Nunes Amaral. High-reproducibility and high-accuracy method for automated topic classification. *Physical Review X*, 5(1):011007, 2015.

[11] Thomas K Landauer, Peter W Foltz, and Darrell Laham. An introduction to latent semantic analysis. *Discourse Processes*, 25(2-3):259–284, 1998.

[12] Victor Lavrenko and W Bruce Croft. Relevance based language models. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 120–127. ACM, 2001.

[13] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th International Conference on World Wide Web*, pages 661–670. ACM, 2010.

[14] Christopher D Manning, Prabhakar Raghavan, Hinrich Schütze, et al. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge, 2008.

[15] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[16] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543, 2014.

[17] Alexander M Petersen, Woo-Sung Jung, Jae-Suk Yang, and H Eugene Stanley. Quantitative and empirical demonstration of the matthew effect in a study of career longevity. *Proceedings of the National Academy of Sciences*, 108(1):18–23, 2011.

[18] Martin F Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[19] Joseph John Rocchio. Relevance feedback in information retrieval. 1971.

[20] Darrell Shakhnarovish. Indyk, nearest-neighbor methods in learning and vision, 2005.

[21] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(2579-2605):85, 2008.

[22] Chong Wang and David M Blei. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 448–456. ACM, 2011.

[23] Takashi Yoneya and Hiroshi Mamitsuka. Pure: a pubmed article recommendation system based on content-based filtering. *Genome informatics*, 18:267–276, 2007.