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

Titipat Achakulvisut1,\*, Daniel E. Acuna1,2 , Tulakan Ruangrong3, Konrad Kording1,2

1 Northwestern University, 2Rehabilitation Institute of Chicago, 3Mahidol University

**Abstract**

An important part of being a professional scientist is to be able to keep a current account of the scholarly material. This task, however, has become increasingly impossible to perform manually due to the large number of publications produced. We could improve this process by adapting algorithms that are known to work relatively well for music and movie recommendations. However, how these algorithms work with scholarly material has largely been unexplored. Here, we develop a Python library that implements a recommendation system based on the content of voted relevant and irrelevant articles. The library provides real-time and accurate suggestions of new articles based on these votes by using a large scale nearest neighbor search. We tested the library on 15K posters from the Society of Neuroscience Conference 2015. We tuned the algorithm to provide suggestions that were closed in topic classifications to human curated poster sessions. We show that our algorithm significantly outperforms suggestions based on keywords. The work presented here promises to make the exploration of scholarly material faster and more accurate than what it is possible today. We also discuss applications to other domains such as figures.

# Introduction

Since the inception of Internet and large online businesses, recommendation systems are routinely used to suggest new items to users based on their past preferences. These systems have been proven useful for music, movies, news, and retail in general [1]. To find new scientific literature, researchers rely mostly on author-provided keywords and citations. These sources of information hinder their exploration because they are known to be poor indicators of novelty and may be potentially biased [2]. Moreover, this problem is pronounced during conferences where appropriate keywords may not even exist, let alone citations. An application of recommendation systems to 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). Many such system are mostly available for commercial software, such as news [3], movie [4], and music [5] applications. There are also more specific systems for scientific literature search. In [6], the authors present a content-based recommendation system that works on PubMed datasets. In [7], the *Scienstein* system combines a large set of criteria for providing literature recommendation. In [8], the authors present a topic-based recommendation system based on a Latent Dirichlet Allocation (LDA) model. It is unclear, however, how these systems scale and the test of their performance is generally indirect.

Here we introduce *Science Concierge* (<http://www.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) [1]. For the recommendation part, it pairs the Rocchio Algorithm [9] with a large-scale approximate nearest neighbor search based on ball trees [10]. The library aims at providing responsive content-based recommendations using only user’s votes rather than collaborative filtering. The Scholarly software, then, provides an open source solution to content-based scientific discovery.

We tune and test the algorithm on a collection of scientific posters from the largest Neuroscience conference in the world, Society for Neuroscience (SfN) 2015. First, we cross validate the LSA model to capture most of the variance contained in the topics. Second, we tune the parameters of the algorithm to recommend posters that maximally resemble human curated classifications into poster sessions. We showed that our algorithm significantly outperforms a popular alternative based on keywords, refining suggestions as the system learns more from the user. A front-end interface that uses the algorithm in the back end is available at [http://www.scholarfy.net](http://sf.scienceofscience.org), where we use data from Society for Neuroscience (SfN) conference ([http://www.sfn.org](http://sfn.org)).

# Materials and methods

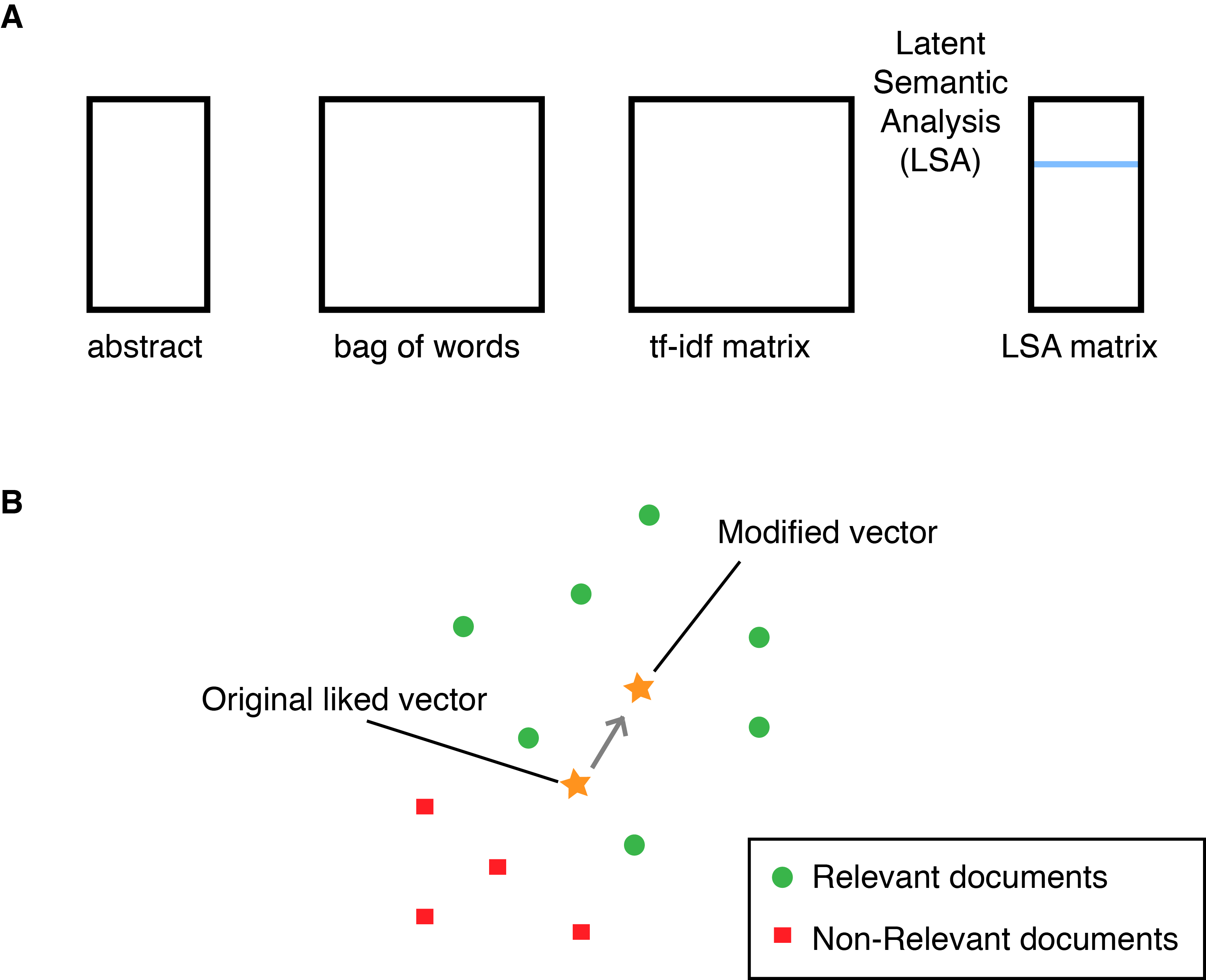
## Conference dataset

Conferences are events that need a rapid understanding of trends. The time from submission, acceptance, and presentation are 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 a conference.

We obtained a license from the Society for Neuroscience (SfN) on the Neuroscience 2015 conference. This is the largest conference in Neuroscience in the world. This dataset included 14,718 posters and talks distributed around 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.

## Content-based recommendation of scientific documents

There are three main design principles behind *Science Concierge* based on the idea that it should work well for scientific literature.First, it aims at using the content of the documents rather than collaborative filter to avoid the Mathew effect in recommendation systems [11]. 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 valid, which means that we will validate suggestions using some external input. Below we describe the methods to achieve these three goals



Macintosh HD:Users:titipat:Desktop:Git:scholarfy:article:figures:component_visualization.pdf

Fig. 1 **Vector representation of documents** (A) Schematic of the workflow for converting abstracts into vector representations (B) Schematic of Rocchio Algorithm (C) 2D dimensionality reduction using t-SNE of abstract vectors, color coded by human curated topics.

### Text preprocessing

As we are mostly applying this algorithm to text documents, the pre-processing steps will be explained in that context. Documents are tokenized removing English stop words and stemming using Porter Stemming algorithm [12]. The terms in the documents are the uni- and bi-grams. This term document matrix is transformed by a term-frequency inverse document-frequency (tf-idf) function where terms that appear fewer than 3 times were removed, and terms that have tf-idf greater than 0.8 were removed.

### Latent Semantic Analysis

The tf-idf matrix is transformed using Latent Semantic Analysis (LSA) to reduce noise and improve smoothness. LSA is a simple topic modeling technique based on singular value decomposition [13]. This technique transforms the document term matrix or, in our case, the sparse tf-idf matrix Xas product of a left singular vector (U), a diagonal singular value matrix (S) and a right singular vector (V) as X = USV’, where diagonal singular values in S are ranked from highest to lowest. The number of vectors to choose depends on how accurate we want to capture the matrix X. This number will be chosen based using cross validation.

The pre-processing is relatively fast on purpose because the algorithm needs to rapidly adapt to changes in the conference. Even with simple level of preprocessing provided by LSA, we can already start understanding 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) ([14], Fig. 1B), coloring each poster with the topical area described in the program (from A through G). Some topics are more clearly separated than others.

### Poster representation based on keywords

The dataset used here also contained keywords. Search by keyword is a popular way used by scientists to discover topics at conferences and use it as a based comparison. 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.

### Rocchio Algorithm and Nearest Neighbor Assignment

The Rocchio algorithm is used to produce recommendations based on relevant and non-relevant document previously voted by the user [15]. It computes a vector of the user’s preferences based on a set of previously voted relevant and non-relevant documents. The nearest neighbors of such resulting preference vector would be documents to be suggested.

Mathematically, given a set of documents vectors that are found to be relevant by the user and another set of non-relevant documents , the Rocchio algorithm finds a document vector that combines both types of documents

|  |  |  |
| --- | --- | --- |
|  |  | (1) |

where 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 and 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 and therefore the mean article replaces this term. The parameters and 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 rely on an approximated approach based on ball trees, which tradeoffs speed and accuracy relatively well [10]. If more documents are added, then the nearest neighbor approach could be approximated even more to scale well, or similarly, if more accuracy is needed, then a more accurate but lower nearest neighbor approach could be used.

## A measure of topic distance to 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 assume that attendees 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 of the suggestion’s topic to the topic of the liked posters, using as distance measure the lowest common ancestor in the topic tree [16]. 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.

# Results

## Parameter optimization

### Number of components for LSA

One parameter of the LSA algorithm is the number of components of the SVD [17], 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. 3A, see Methods). In this way, we were able to find that the appropriate number of components was around 100.

Macintosh HD:Users:titipat:Desktop:Git:scholarfy:article:figures:performance_vs_components.pdf

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 provides suggestions using two sets of documents, one found to be relevant and another found to be non-relevant. The algorithm differently weighs the importance of relevant documents with a parameter and the importance of non-relevant with a parameter (see Eq. 1). We tuned the parameters and 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 and parameters and computed the average topic distance over a set of 1,000 simulations (Fig. 4).

We found that voting as non-relevant those posters that are away 1 distance produces large effects on performance. Interestingly, we found that the parameter makes the performance decrease almost always. 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 alues for the parameter are always larger than 1. Additionally, we found that non-relevant posters that away 3 distance in topic space have little effect on the suggestions (Fig. 4C). 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 8 and .

Macintosh HD:Users:titipat:Desktop:Git:scholarfy:article:figures:performance_vs_components.pdfMacintosh HD:Users:titipat:Desktop:Git:scholarfy:article:figures:performance_vs_components.pdfMacintosh HD:Users:titipat:Desktop:Git:scholarfy:article:figures:performance_vs_components.pdf

Fig. 3 **Finding best parameters to weight 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

## 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 Methods).

For each of the algorithms tested below, we perform 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. 5, Random line) 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 another common recommendation technique that relies on using human curated “tags” for each document. In our dataset (see Materials and Methods), 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. Therefore, we model keyword counts using a tf-idf to 30 SVD dimensions. Preliminary analysis revealted that 30 dimensions was an appropriate number to capture most of the variability. Suggestions using keywords performs significantly better than random suggestions (paired *t*(4999) = 74.74, *p* < 0.0001) and keyword suggestion performance improves with the number of votes (Fig. 5, *t*(4998) = -4.428, *p* < 0.0001). While keywords allows for better suggestions than the null model, authors need to manually provide them.

Scholar Concierge uses the abstracts to automatically extract topics and provide suggestions. We found that Scholar Concierge produces significantly better suggestions than keywords (Fig. 5, *t*(4999) = 78.95, *p* < 0.0001). Also, we found that the performance significantly improves as the system learns more about a user (*t*(4998) = -31.6, *p* < 0.001). Scholar Concierge thus does not require humans to provide keywords and improves the recommendation as more votes are provided.

Macintosh HD:Users:titipat:Desktop:Git:scholarfy:article:figures:performance_vs_votes.pdf

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 their suggestions. Scholarfy significantly improves on keywords and event further with more votes

# Discussion and conclusion

Discovering new and relevant scholarly material is almost impossible without the use of systems that help scientists find them. 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 our source code of the system so others can expand its functionality. In sum, this article presents a system that can make scientific discovery faster and more accurate than what is possible today.

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 system degraded. If those non-relevant posters were far away, then the degradation disappeared. For the ranges of parameters tried here, the non-relevant posters are better used as a way for the user to remove recommendations from the list of suggested posters. In the future, we will expand the experiment to include real world data on more posters that were relevant. This may allow the algorithm to better understand the topic preference vector better 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 time 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 [18, 19], which recently have been shown to scale well [20]. To better capture entities embedded in the text, future research will also investigate how to use deep learning modeling of words and phrases [21, 22]. However, it is unclear how many of the more sophisticated modeling of topics or word embedding provides significant improves while coping with scalability. Our system may already provide an appropriate level of speed and accuracy for our application.

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 [23]. With long term users that provide hundreds or even thousands of votes, it may be more accurate to cast the recommendation problem as one of classification [24]. 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 make their algorithms or corpora openly available, making them almost impossible to study. By opening the way in which we suggest items (i.e., <http://www.github.com/titipata/science_concierge>), we will engage the scientific community to collaborate on our source code base. 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, datasets, and others). In sum, our systems provide an open and fast approach to accurately discover new research questions and promises to accelerate science in many ways.

**Acknowledgements**

Titipat Achakulvisut was supported by Royal Thai Government Scholarship grant #50AC002. DAniel E. Acuna was supposed by the John Templeton Foundation grant #. Tulakan Ruangrong would like to thank… Konrad Kording was supposed by NIH # and John Templeton Foundation #. Thanks to SfN for providing the abstract dataset for testing.

**Author Contributions**

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

**References**

1. Manning, C.D., P. Raghavan, and H. Schütze, *Introduction to information retrieval*. 2008, New York: Cambridge University Press. xxi, 482 p.

2. Chavalarias, D. and J.P. Ioannidis, *Science mapping analysis characterizes 235 biases in biomedical research.* J Clin Epidemiol, 2010. **63**(11): p. 1205-15.

3. Li, L., et al. *A contextual-bandit approach to personalized news article recommendation*. in *Proceedings of the 19th international conference on World wide web*. 2010. ACM.

4. Bell, R.M. and Y. Koren, *Lessons from the Netflix prize challenge.* ACM SIGKDD Explorations Newsletter, 2007. **9**(2): p. 75-79.

5. Ali, M., C.C. Johnson, and A.K. Tang. *Parallel Collaborative Filtering for Streaming Data*. 2011; Available from: <https://github.com/MrChrisJohnson/CollabStream>.

6. Yoneya, T. and H. Mamitsuka. *PURE: a PubMed article recommendation system based on content-based filtering*. in *Genome Inform*. 2007. World Scientific.

7. Gipp, B., J. Beel, and C. Hentschel. *Scienstein: A research paper recommender system*. in *Proceedings of the international conference on emerging trends in computing (ICETiC’09)*. 2009.

8. Wang, C. and D.M. Blei. *Collaborative topic modeling for recommending scientific articles*. in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2011. ACM.

9. Rocchio, J.J., *Relevance feedback in information retrieval.* 1971.

10. Shakhnarovich, G., T. Darrell, and P. Indyk, *Nearest-Neighbors methods in Learning and Vision: Theory and Practice*. 2006: MIT Press.

11. Petersen, A.M., et al., *Quantitative and empirical demonstration of the Matthew effect in a study of career longevity.* Proc Natl Acad Sci U S A, 2011. **108**(1): p. 18-23.

12. Porter, M.F., *An Algorithm for Suffix Stripping.* Program-Automated Library and Information Systems, 1980. **14**(3): p. 130-137.

13. Landauer, T.K., P.W. Foltz, and D. Laham, *An introduction to latent semantic analysis.* Discourse Processes, 1998. **25**(2-3): p. 259-284.

14. Van der Maaten, L. and G. Hinton, *Visualizing data using t-SNE.* Journal of Machine Learning Research, 2008. **9**(2579-2605): p. 85.

15. Rocchio, J., *Relevance feedback in Information Retrieval*, in *The SMART Retrieval System: Experiments in Automatic Document Processing*, Salton, Editor. 1971, Prentice-Hall.

16. Aho, A.V., J.E. Hopcroft, and J.D. Ullman, *On finding lowest common ancestors in trees.* SIAM Journal on computing, 1976. **5**(1): p. 115-132.

17. Bishop, C.M., *Pattern recognition and machine learning*. Information science and statistics. 2006, New York: Springer. xx, 738 p.

18. Blei, D.M., A.Y. Ng, and M.I. Jordan, *Latent dirichlet allocation.* the Journal of machine Learning research, 2003. **3**: p. 993-1022.

19. Lancichinetti, A., et al., *High-Reproducibility and High-Accuracy Method for Automated Topic Classification.* Physical Review X, 2015. **5**(1): p. 011007.

20. Hoffman, M., F.R. Bach, and D.M. Blei. *Online learning for latent dirichlet allocation*. in *advances in neural information processing systems*. 2010.

21. Mikolov, T., et al., *Efficient estimation of word representations in vector space.* arXiv preprint arXiv:1301.3781, 2013.

22. Pennington, J., R. Socher, and C.D. Manning, *Glove: Global vectors for word representation.* Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014), 2014. **12**: p. 1532-1543.

23. Friedman, J., T. Hastie, and R. Tibshirani, *The elements of statistical learning*. Vol. 1. 2001: Springer series in statistics Springer, Berlin.

24. Lavrenko, V. and W.B. Croft. *Relevance based language models*. in *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*. 2001. ACM.