**Latent Semantic Analysis for Retrieving Related Biomedical Articles**

by

Sheng-Ting Lin

B.Sc., McGill University, 2014

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

in

THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

(Bioinformatics)

THE UNIVERSITY OF BRITISH COLUMBIA

(Vancouver)

April 2017

© Sheng-Ting Lin, 2017

# Abstract

Retrieving relevant scientific papers in a scalable way is increasingly important, as more and more studies are published. PubMed’s relevant article recommendation is based on MeSH assignments by indexers, which requires significant human resources and can become a limitation in making papers searchable. Many recommendation systems use singular value decomposition (SVD) to pre-compute related products. In this study, we look at using latent semantic analysis (LSA), an application of SVD to determine relationships in a set of documents and terms, to find related biomedical papers. We focused on determining the best parameters for SVD in retrieving relevant biomedical articles given a paper of interest. Using PubMed's recommendations as guidance, we found that using cosine distance to measure document similarity leads to better results than using Euclidean distance. We re-evaluated other parameters, including the weighting scheme and the number of singular values and using a larger abstract corpus. Finally, we asked people to compare the relevant abstract retrieved with our method against those retrieved by PubMed. Our method retrieved sensible articles that were chosen over PubMed's relevant papers one-third of the time. We looked into the abstracts retrieved by either method and discuss possible areas for experimentation and improvement.

# Preface

As part of the core curriculum in the Bioinformatics Training Program, I undertook a rotation project in the laboratory of Dr. Steven Jones. Working closely with graduate student Jake Lever, the experience from the rotation gave me the inspiration to the core idea and much background knowledge of this thesis project, including dimensionality reduction with SVD and text mining of biomedical literature. Moreover, the infrastructure, such as the compute environment, that was already in place from the rotation project has helped me progress through my thesis efficiently.

Jake Lever created the word list and the code for tagging the biomedical terms. Sita Gakkhar developed the program that calculates the ranks for papers in the 13-million abstract matrix. I did the analysis presented in this study. The materials in this thesis have not been published.

# Table of Contents

Abstract ii

Preface iii

Table of Contents iv

List of Tables vi

List of Figures vii

List of Abbreviations ix

Acknowledgements x

Dedication xi

Chapter 1: Introduction 1

1.1 Previous work and motivation 1

1.2 Types of recommender systems 3

1.3 SVD: Matrix factorization in collaborative filtering 9

1.4 Latent Semantic Analysis (LSA) 14

1.5 PubMed and other article recommenders 17

Chapter 2: Parameters for retrieving relevant abstracts 21

2.1 Overview 21

2.2 Methods and tools 21

2.2.1 Obtaining biomedical abstracts 23

2.2.2 Text processing 24

2.2.3 Singular value decomposition and PowerGraph (GraphLab v2) 26

2.2.4 Calculate relevant documents 29

2.2.5 Evaluation: precision and singular values analysis 30

2.3 Result 31

2.4 Discussion 35

Chapter 3: Sampling word occurrences 36

3.1 Overview 36

3.2 Method 38

3.3 Result 39

3.3.1 Term frequency matrix sampling 39

3.3.2 TF-IDF matrix sampling 40

3.3.3 Sampling on the 13 million abstract corpus 44

3.4 Discussion 48

Chapter 4: Evaluation with rankings and surveys 49

4.1 Overview 49

4.2 Method 49

4.2.1 Rank analysis 49

4.2.2 Survey analysis 50

4.3 Result 53

4.3.1 Rank analysis 53

4.3.2 Survey analysis 54

4.4 Discussion 58

Chapter 5: Discussion 61

Bibliography 64

# List of Tables

Table 2.1 Parameters for document retrieval 21

Table 4.1 Number of relevant papers with rankings < 10,000 53

Table 4.2 Related papers with low ranks in SVD 53

Table 4.3 Result from Survey 56 54

Table 4.4 Survey result 55

Table 4.5 Abstracts from PubMed were considered more relevant 56

Table 4.6 Abstracts from SVD were somewhat related to the targeted papers 56

Table 4.7 Abstracts from PubMed and SVD were considered equally relevant or irrelevant 57

Table 4.8 Abstracts from SVD were considered more relevant 58

# List of Figures

Figure 1.1 Examples of finding new associations 2

Figure 1.2 Computing predictions using feature vectors 5

Figure 1.3 Wikipedia's illustration on collaborative filtering 7

Figure 1.4 An Amazon product page 8

Figure 1.5 Full Singular value decomposition (SVD) and reduced SVD. 11

Figure 1.6 Reduced SVD and decomposed matrices. 12

Figure 1.7 PubMed - similar articles 18

Figure 2.1 Data and tools for testing different parameters on retrieving biomedical abstracts 22

Figure 2.2 Making an abstract corpus 23

Figure 2.3 Bag of words model for biomedical literature 25

Figure 2.4 PowerGraph's distributed parallel computation 27

Figure 2.5 Precision of the retrievals by SVD 30

Figure 2.6 Average precision over top 10 abstracts for 92 different matrices 32

Figure 2.7 Maximum precisions for the 92 abstract corpora 32

Figure 2.8 Average error estimates from 92 abstract corpora 33

Figure 2.9 Average singular values from the 92 abstract corpora 33

Figure 2.10 Average coverage of the 92 abstract corpora 34

Figure 2.11 Running time with different distance metric 34

Figure 3.1 Random sampling on a matrix 37

Figure 3.2 Coverage values at different NSV and sampling rate for 10 TF matrices 39

Figure 3.3 Error estimates of singular values for TF matrices 40

Figure 3.4 Coverage for TF-IDF forms of sampled matrices 41

Figure 3.5 Coverage for TF-IDF matrix scaled by 10 and then sampled 42

Figure 3.6 Coverage for TF-IDF matrices scaled by 100 and then sampled 43

Figure 3.7 Coverage for TF-IDF matrix scaled by 1000 and then sampled 43

Figure 3.8 Coverage of the sampled and original 13M-abstract TF matrices 44

Figure 3.9 Singular values of the sampled and original 13M-abstract TF matrices 45

Figure 3.10 Error estimate of the sampled and original 13M-abstract TF matrices 45

Figure 3.11 Coverage of the sampled and the original 13M-abstract TF-IDF matrices 46

Figure 3.12 Singular values of the sampled and original 13M-abstract TF-IDF matrices 47

Figure 3.13 Error estimates of the sampled and original 13M-abstract TF-IDF matrices 47

Figure 4.1 Question template for a survey 51

# List of Abbreviations

* LSA – Latent semantic analysis
* SVD – Singular value decomposition
* NLM – National Library of Medicine
* MeSH – Medical Subject Headings
* UMLS – Unified Medical Language System
* TF-IDF – Term frequency and inverse document frequency
* TF – Term frequency
* NSV – Number of singular values
* TP  - True positive
* FP – False positive

# Acknowledgements

I would like to thank my supervisor, Dr. Steven Jones, for his mentorship and guidance through both the rotation project and this thesis study. He gave me the liberty to develop my ideas and the resources I needed to complete the study. His timely critiques during our biweekly meetings and weekly lab meeting helped me progress and learn through the project.

I want to also thank my supervisory committee members, Dr. Sara Mostafavi and Dr. Inanc Birol, for their time and valuable insights. I also thank Dr. Paul Pavlidis for taking the time to chair the examining committee. I want to give special thanks to Sharon Ruschkowski, the Bioinformatics program coordinator, for being the first contact throughout my time in the graduate program.

I would like to thank Jake Lever, one of the lab members, for guiding me through my rotation project and supporting my thesis with his reusable codes and expertise on the subject. Our coffee sessions and numerous email exchanges were tremendous help.

I would also like to thank Sita Gakkhar, a critical person in this project, for his valuable ideas and assistance. His expertise has helped me understand dimensionality reduction much better during the rotation project. The TrackRank program he wrote had made it possible to do the rank analysis much more efficiently. His idea on sampling matrices also gave rise to Chapter 3.

Finally, I want to thank all my lab members for being there listening to my progress updates during the big data meetings. Moreover, I cannot give enough thanks to my peers in the Bioinformatics Training Program who invested time and energy in doing my surveys.

# Dedication

*To my parents, Donna and Gauss  
my brother, Kevin*

## Introduction

### Previous work and motivation

The concepts behind using latent semantic analysis to retrieve related biomedical abstracts span across several fields, including knowledge discovery, dimensionality reduction in recommendation systems and text retrieval.

Advances in computational and biological methods, such as microarrays, sequencing and other large-scale experimental methods, have led to an unprecedented growth in the publications and biomedical data [[1]](https://paperpile.com/c/CvqbpE/fGjoS). Almost every known gene and protein are reported somewhere in the biomedical literature and other forms of records, which sometimes exist in rather isolated fields of expertise [[2]](https://paperpile.com/c/CvqbpE/ScRJO). Effectively surveying the biomedical literature can help identify previous unappreciated similarities and linkages.

In 1986, Swanson presented the ABC model and hypothesized the possible link between Fish oil and Raynaud's disease [[3]](https://paperpile.com/c/CvqbpE/0mjse). He collected 1000 articles on fish oil, some of which discussed fish oil's effect of lowering blood viscosity and inhibiting platelet aggregation. He also found another 2000 articles on patients who have Raynaud's disease who have increased blood viscosity and aggregated platelet (Figure 1.1). It was the first reported literature-based model that used shared attributes to link two isolated disciplines, A and B, together. Since then, there has been a strong interest in the research community do literature-based discovery, or "mining" the biomedical literature to generate novel hypotheses [[1]](https://paperpile.com/c/CvqbpE/fGjoS). These tasks range from predicting specific information such as a proteins cellular location to attempts like constructing gene networks by examining the co-occurrences of gene names in publicly available literature and databases [[4,5]](https://paperpile.com/c/CvqbpE/EgtGW+vXRTL). Literature-based discovery can also be used to find other relationships such as a gene-disease or drug-disease relationship in order to help identify novel therapeutic targets and repurpose drugs [[2]](https://paperpile.com/c/CvqbpE/ScRJO).

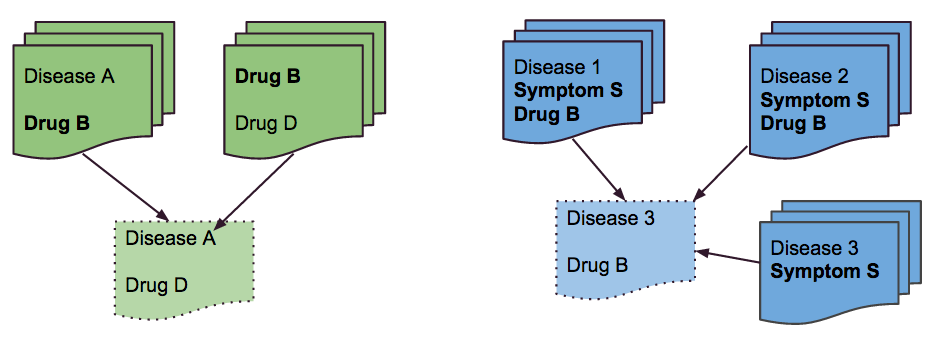
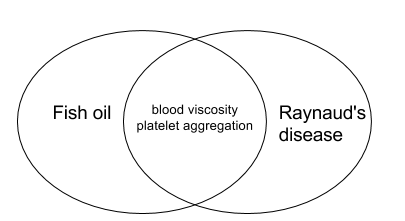


Figure 1.1 Examples of finding new associations

As shown in Figure 1.1, knowledge discovery is about finding implicit associations between previously unrelated concepts. The example on the left shows the Swanson's ABC model which used shared information between two discipline, studies on dietary fish oil and symptoms of Raynaud's disease, to hypothesize that fish oil might be beneficial to patients with Raynaud's disease. However, there are other types of indirect links that would not fit into the ABC model. In the example on the right, symptom S and Drug B are mentioned together in two different contexts: Disease 1 and Disease 2. If there is a third disease being discussed together with S, there might also be some relationship between Drug B and Disease 3. Knowledge discovery is about finding these unknown relationships by connecting these known data structures together.

One interest of our research group is leveraging machine learning techniques to do literature-based discovery to identify previously unknown linkages between biomedical concepts, such as finding novel gene-disease and drug-disease associations. The goal is to find implied linkages between two biomedical concepts that are never mentioned together in the same documents by surveying numerous articles.

This challenge is analogous to making recommendations to people when there are also a great number of items and users. In the context of recommendation systems in e-commerce, there are many associations between items and users, such as people's purchasing history and ratings with different products. Recommendation systems need to make a new association between a user and a product or, in other words, predict that someone would be interested in an item, by looking at patterns in that information.

We had previously applied a technique used in recommendation system on biomedical term co-occurrence data and found links between non-associated terms. When we sought to explain what led to those new concepts to become associated together, we came across an application in natural language processing that used term co-occurrence to measure word similarity and thus identify synonyms [[6]](https://paperpile.com/c/CvqbpE/Y49Aw). We hypothesized that scientific documents could be measured in a similar way. In other words, similar biomedical papers are those that contain related concepts, and the relatedness is inferred from the main patterns in the term co-occurrences in the papers. This approach of representing document has been applied in other text retrieval applications due to the many problems that it can address [[6,7]](https://paperpile.com/c/CvqbpE/oXPGi+Y49Aw), as discussed later in the chapter. We wanted to see if this technique could be used to retrieve related biomedical papers.

One most well known text retrieval application for biomedical paper is perhaps PubMed's search engine, which recommends articles related to the one that users are currently looking at. It uses manually assigned MeSH (Medical Subject Headings) terms that define each paper to retrieve related articles. Given that scientific papers are being published at a growing rate, we thought it is worthy to look into alternatives that do not rely on human indexers.

This chapter presents a literature review on the related concepts for this thesis, including recommendation system, PubMed, SVD, and latent semantic analysis. The terms "paper", "articles" and "documents" are used interchangeably.

### Types of recommender systems

One aim of this thesis is to propose a technique of recommendation system to retrieve and recommend related biomedical papers. Therefore, it is important to survey the principle, methods, and evaluation of recommendation systems that are known today.

Recommender systems select a small set of items or manageable amount of content from a larger pool information such that the people find the selection relevant to their needs. They represent decision-making strategies that come in many forms, including non-personalized versus personalized recommenders and non-collaborative versus collaborative systems.

Recommendation systems of various forms have existed for a long time to help people making choices. They started off as purely manual and have become more automatic and sophisticated to meet the growing demand of e-commerce. Historically, people relied on an acquaintance’s recommendation or traditional media to decide which movie watch or make a purchase decision. This system has some limitations because people might disagree with others' movie taste or news editors' opinions. Nowadays, the internet has made it possible for such services to source many more opinions on a greater number of products.

One early adoption is the non-personalized systems, which recommend the same item to different people [[8]](https://paperpile.com/c/CvqbpE/NuSKq). These recommendations are usually selected from the opinions of many people in the forms of ratings or reviews. The quality of this class of recommender depends on the algorithm that aggregates opinions, sometimes putting more weight on the recent ones, and ranking selections [[9]](https://paperpile.com/c/CvqbpE/4fZqb). However, the collected opinion of a community does not always align with what each individual would prefer [[9]](https://paperpile.com/c/CvqbpE/4fZqb). Non-personalized systems can be improved by considering information about a specific group of people and their general product preferences. For example, young adults can have a different purchasing pattern than the seniors. E-commerce merchants can use the information to construct an automatic non-personalized recommendation system. Grundy was one of the first non-personalized recommender systems that utilized the capacity of computers in 1979 [[10]](https://paperpile.com/c/CvqbpE/N4Fxn). It was a computer-based librarian that modeled users as individuals with different goals and used hard-coded stereotyped book preferences to generate recommendations.

To make an effective recommendation, a recommendation system must be further tailored. Store clerks greet customers by asking what they're looking for, and travel agents consider the customer's budget and preferences before suggesting a perfect trip. These are the earliest forms of manual and personalized recommendation systems that allowed people to focus on what they like instead of exploring all the options. We can also read reviews and make a personalized recommendation to ourselves. However, not every medium or product types have reviews written by experts or have sale representatives. We now rely on a more automated way to consider both people's preferences and items' properties to recommend candidate items. The content-based recommenders tag products with some measurable features that allow consumers to filter products [[11]](https://paperpile.com/c/CvqbpE/Jgiep). For a restaurant, it could be the price range and the quality of service. People can indicate what they value about a restaurant, and the recommendation system can compute a predicted score for each restaurant and recommend the ones with the highest scores.

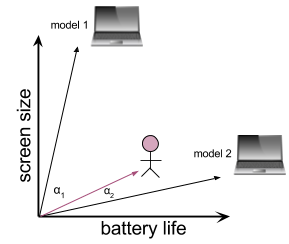


Figure 1.2 Computing predictions using feature vectors

Predicting whether someone would like an item more than the other is analogous to finding the product whose feature vector is closest to the customer's preference vector. In this example shown in Figure 1.2, if a customer cares much more about the battery life than the size of the screen of a laptop, we would recommend a laptop that has a longer battery life at the expense of larger screen size. To do this, we can compute some distance between the customer’s desired model and the existing models, which can be the angles α1 and α2 and recommend model 2 to the customer because of α2  < α1. This idea can be extended to more than two features and multiple laptops or other products, like movies, that can be described by some quantifiable features. After calculating the predictions for each product relative to a customer's preference, the system can then rank products on the predicted scores and recommend those that are on the top of the list.

Sometimes people do not completely know their inherent preferences and how much they value different attributes. Sometimes a personalized recommendation system can still build the personal profile of each individual by soliciting explicit ratings and sometimes non-explicit ratings [[12]](https://paperpile.com/c/CvqbpE/yYbuI). Explicit ratings come directly from the people’s feedback after watching a video or using a product by giving stars or a number, say from 1 to 10. Implicit ratings come from people's action and behaviour on the items. For example, if a person watches a five-minute video for 5 seconds before quitting the page, the system could infer that they did not enjoy the video.

On the other hand, collaborative recommenders, or recommender systems that use collaborative filtering techniques, make predictions (e.g. filtering) about one individual's preferences by using others' preferences [[13]](https://paperpile.com/c/CvqbpE/KsnSi). A decade after Grundy, collaborative filtering arose as a solution to help people search for relevant information in a large space of information, including emails. Tapestry, one of the earliest collaborative filtering system, was an experimental mail system that crowd-sourced users' opinions on content to deliver relevant information to people [[14]](https://paperpile.com/c/CvqbpE/gBJoD). It was part basic content filtering of the text and part manual collaborative filtering system that involves human to annotate their reaction to the e-mails.

As illustrated in Figure 1.3, the underlying assumption in collaborative filtering is that if people agree on the relevance of some items, then they are likely to agree on the other items. To predict whether someone would like a product, we select the users who have made similar choices to our target individual. We then aggregate the opinions of those people to predict whether our user of interest would like a product. In this scenario, the prediction labelled with the question mark would be that they would not like the video because the two other people most similar to them have disliked it. The binary choices of thumbs-up and thumbs-down can also be replaced by numerical ratings in other applications.

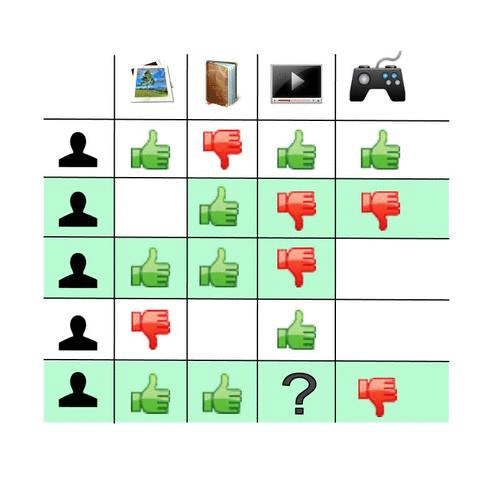
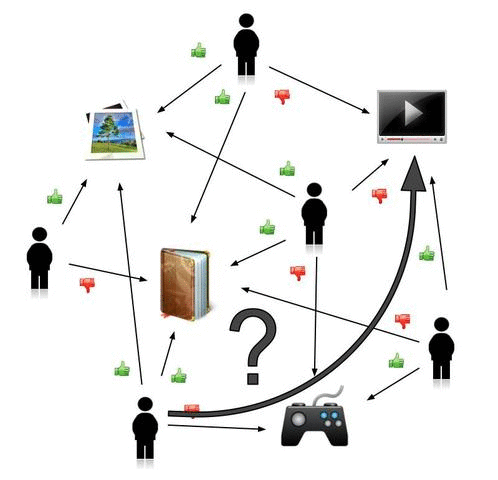


Figure 1.3 Wikipedia's illustration on collaborative filtering

GroupLens, one of the first automatic collaborative-filtering tools, soon came along in 1994 [[15]](https://paperpile.com/c/CvqbpE/BXgwG). It helped people find the readings they would like among many available articles using the heuristics that people who exhibited similar preferences would likely to agree on new articles. Users only need to provide ratings and do some measurable actions. In the late 1990s, many commercial adaptations of recommender systems emerged in an effort to increase sales, improve customer experience, and help people discover new products that they might not seek out otherwise [[13]](https://paperpile.com/c/CvqbpE/KsnSi). Perhaps one well-known example is Amazon.com, which uses collaborative filtering to learn what users might like and recommend the products and what products are similar to each other by looking at their past purchasers. As shown in Figure 1.4, Amazon's product page recommends items by showing products that are often bought together with the product in the view. Under "Customers who bought this item also bought" heading, there are books about data science, statistics, machine learning or a combination of these topics. If we had used the content or attributes of this book to find relevant items instead of user's purchasing history, we would likely retrieve some other Python machine learning books and that might not be as helpful to someone who likely has more diverse interests..

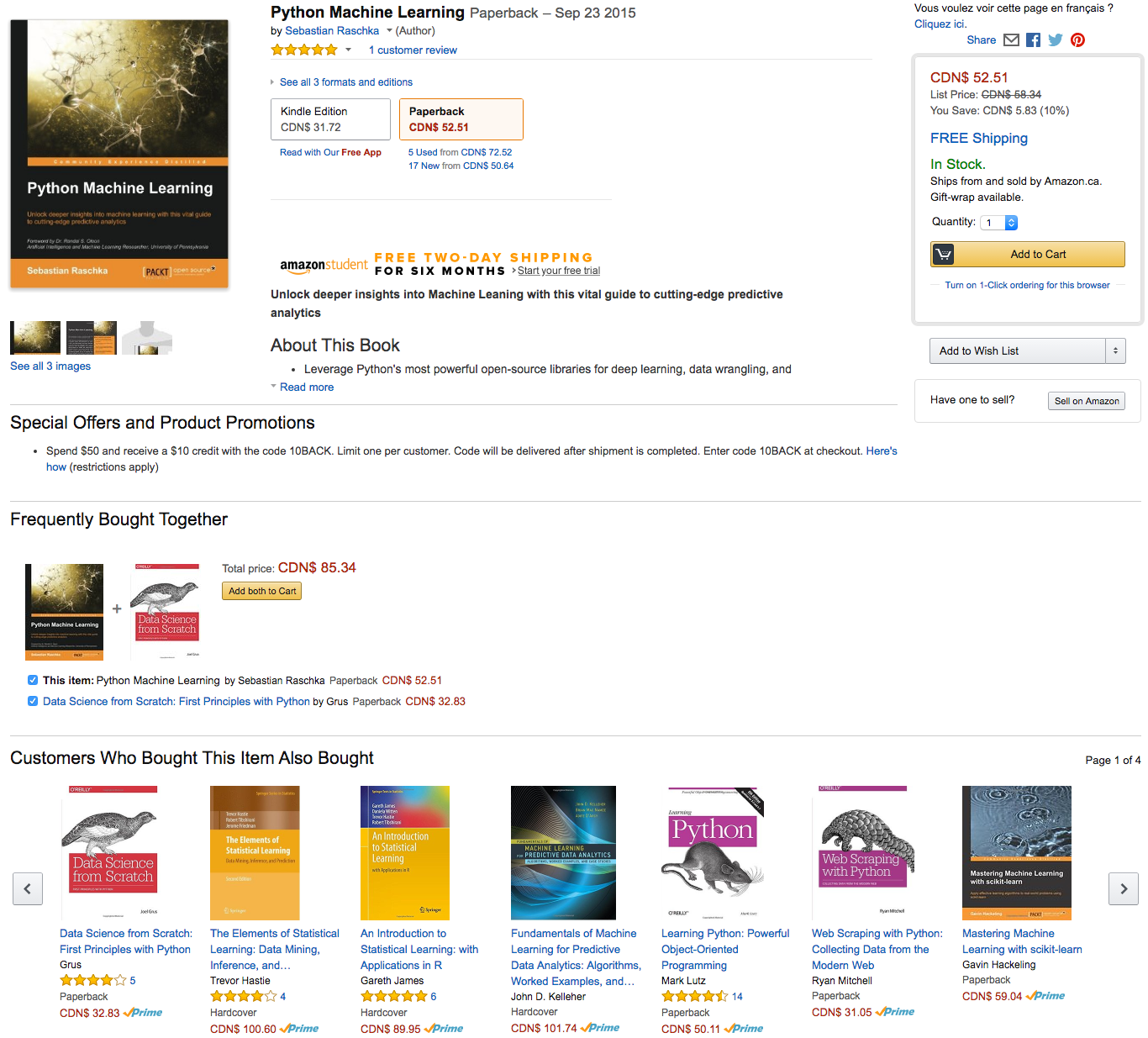


Figure 1.4 An Amazon product page

While there is a strong business incentive to improve the adaptations of recommendation systems, many had also started to apply the algorithms and related techniques to other areas including education, medicine, and more. For instance, OWL (Organization-Wide Learning), developed by Mighter Corporation, was one of the first recommender systems that integrate continuous knowledge acquisition in a tool that tailors to each individual's specific needs [[16]](https://paperpile.com/c/CvqbpE/htgbH). It was built into a word processor that educate people about the keyboard shortcuts. Based on how people use the word processor, the system assigns a priority score to unused word commands and suggests them to the users so they could type more efficiently [[16]](https://paperpile.com/c/CvqbpE/htgbH).

These are just a few examples and classes of recommendation systems. Each of them can be further adjusted with the fine details of computations that take the user experience and the idiosyncrasies of the data into consideration. Recommenders can also be used in combination as hybrid recommenders to cover more use cases and datasets and compensate the shortcomings of each individual type of recommenders [[12,17]](https://paperpile.com/c/CvqbpE/Vnn0S+yYbuI).

In 2006, Netflix announced a one million dollar prize for the competition of building recommenders that could out-perform their own CineMatch algorithm [[17,18]](https://paperpile.com/c/CvqbpE/J2ZAZ+Vnn0S). The event greatly stirred the interest of academia and hobbyists in recommendation systems. After three years of competition, the prize was awarded to Koren's team that used a hybrid recommender that combines different matrix factorization models together into a collaborative filtering solution [[19,20]](https://paperpile.com/c/CvqbpE/Cug8r+CpiVm). Many of the models used were different types of singular value decomposition (SVD). Not only did the prize amount show how much industry values good recommender systems, the work stimulated great interest and research into the field [[13]](https://paperpile.com/c/CvqbpE/KsnSi). Many top entries, including the winning team, also demonstrated that matrix factorization models are superior to classic nearest-neighbour techniques for doing collaborative filtering to produce product recommendations [[21]](https://paperpile.com/c/CvqbpE/MeCEk).

### SVD: Matrix factorization in collaborative filtering

Matrix factorization is closely tied with the collaborative filtering technique in recommendation systems. It can process large user-item data by extracting patterns from the data and removing noise at the same time.

Collaborative filtering can be subdivided into two types: user-user collaborative filtering (CF) and item-item CF. In user-user CF, people exhibit similar preference as the current user are used to predict whether the current user would like the product [[13]](https://paperpile.com/c/CvqbpE/KsnSi). The data is usually represented as a matrix where rows are users and columns are items, as shown in Figure 1.3. Finding similar users to our target user is essentially the same as finding rows that are closest to the row for target user in the matrix. One of the most critical design decisions for user-user CF is thus the metrics for measuring the similarity between the rows. Commonly used metrics include Pearson correlation, Spearman rank correlation, cosine similarity and more [[13]](https://paperpile.com/c/CvqbpE/KsnSi). Taking a weighted average of ratings of the similar users on each item can then predict the preferences. The items recommended to the user are those with the highest predicted scores.

The item-item CF is similar to the user-user CF except that the roles of users and items are swapped. The recommendation systems measure the similarities between item pair by looking at the purchase records. Two items are more similar if more users have purchased both of them [[13]](https://paperpile.com/c/CvqbpE/KsnSi). This is different from content-based recommenders as discussed earlier, where items with similar features are considered close. In the case of item-item CF, items of similar relevance are more likely to be purchased together. An example of this type of recommendation can be seen on the Amazon's product page (Figure 1.4).

Both the numbers of users and items can be in the order of millions like in the cases of the websites YouTube and Amazon.com. If we represent this data in the form of a user-item matrix as shown in Figure 1.3, it would be an enormous matrix in which each item vector is |U|-dimensional where U is the set of all users, and each user vector is |X| where X is the set of all items. There could also be redundancy in the matrix, because some items are similar (e.g. books of the same title in paper cover or hardcover) and that some users have identical or very similar tastes. In a user-item matrix, items are treated as independent items. If someone watched The Hunger Games (2012) and another person watched The Hunger Games: Catching Fire (2013), the system would not inherently consider these individuals having similar movie preferences based on the obvious similarity of these two movies. We would of course expect that the recommender system to ultimately determine that these movies are highly related.

Can we somehow divide items and users into some groups to reduce the dimensions of the rating space? This is explored in the earlier discussion that some recommendation systems use stereotypes of people or features of items instead of items themselves to make recommendations. However, doing so would require us to somehow come up with some biases and a comprehensive list of feature descriptors for millions of users and items.  Even if we do have a set of measurable features, we might still not be able to account for all possible ways of how someone would consider a product. Suppose we use genres to categorize our movies. If the two people both rated the Hunger Games movies highly after watching them, we would then conclude that the two people have similar taste in that they like action movies. However, our conclusion would be untrue if the person watched The Hunger Games (2012) actually dislikes action movies but would watch anything the actress Jennifer Lawrence starred in.

Many of the top entries of the Netflix Grand Prize used SVD, or singular value decomposition, as their matrix factorization model for collaborative filtering [[20]](https://paperpile.com/c/CvqbpE/CpiVm). SVD would allow us to mathematically aggregate users and items into combinations of user interests and topics of the items based on our observed user-item data [[22]](https://paperpile.com/c/CvqbpE/Mbf5W). First, SVD transforms correlated variables into uncorrelated ones that better expose the relationship in the original data [[23]](https://paperpile.com/c/CvqbpE/JCpRd). While doing so, SVD also orders the dimensions along which data exhibits the most variation so that we can approximate the original data with much fewer dimensions that preserve the main relationships in the data [[23]](https://paperpile.com/c/CvqbpE/JCpRd).

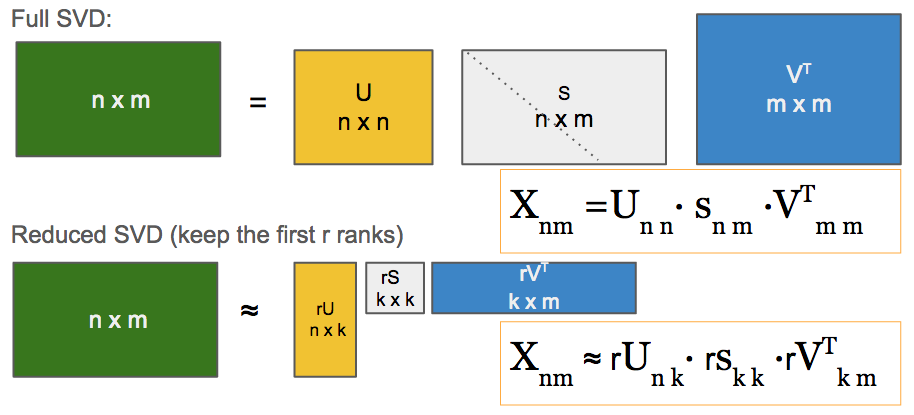


Figure 1.5 Full Singular value decomposition (SVD) and reduced SVD.

SVD is based on the theorem that a rectangular matrix X can be broken down into the product of three matrices: , where columns of U are orthonormal eigenvectors of X XTand columns of V are orthonormal eigenvectors of XTX, and S is a diagonal matrix with singular values in descending order [[24]](https://paperpile.com/c/CvqbpE/R0Gt3). The matrix U represents the row space in X, whereas matrix V represents the column space of X. Singular values taper off and approach zeros at some point, which mean the corresponding columns in U and V do not contribute much to the data. Therefore, we can do an approximation (and hopefully generalization) of our original data with only the first k singular values and their corresponding columns in U and V. This is called k-rank approximation of a matrix in reduced SVD. By doing so, we discard the less significant columns by keeping only the columns where the singular values are large to massively reduce the data while preserving the main relationships within the data. In the case of movie recommender, the original data set is our rating data, and the two variables are users and movies. When we use reduced-SVD to decompose our rating matrix, we identify k topics that are combinations of user's interest in different topics and their extent of liking a certain topic [[13]](https://paperpile.com/c/CvqbpE/KsnSi). Predicting the preference of User *a* for any given item *i* can be done by taking a weighted dot product:

where *Ua*  is the row in *U* corresponding to User *a*, *Vi* is the row in *V* corresponding to item *i* and *sf*  is a singular value.

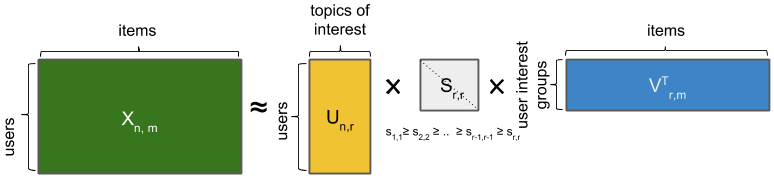


Figure 1.6 Reduced SVD and decomposed matrices.

Reduced SVD compresses a large matrix into smaller and more compact ones and also has the effect to group similar items/users together and remove the noise instead of considering every item and user as one dimension [[22,25]](https://paperpile.com/c/CvqbpE/Mbf5W+7QD0q). This prevents over-fitting because our data, in the context of movie rating, would inevitably have sporadic ratings that fall outside of the main patterns of user preferences [[13]](https://paperpile.com/c/CvqbpE/KsnSi). The best k-rank approximation of the matrix X is the reconstructed matrix Xk from taking the product of the three reduced matrices that keep the top k dimensions, because SVD minimizes the global root-mean-square deviation (RMSE), or Frobenius norm *|| X - Xk ||*F for all rank-k matrices [[13]](https://paperpile.com/c/CvqbpE/KsnSi). Frobenius norm will be discussed in greater detail when we use it as a metric to measure how much information from the matrix is captured within the reduced SVD.

The dimensionality reduction technique, SVD, has some trade-offs that prevent it from becoming widely adopted in every application. First, not all dimensions of the decomposed matrix are describable because each of them can be a collection of different dimensions that contribute to it at various degrees. For example, given a matrix of user by movie attributes matrix, the one of the first few singular value could be something like this:

where F is a set of features that are contributing to the singular value, and n is a real number. Due to the lack of transparency in SVD, it is difficult to generate insight about the patterns in a user-product data by looking at the decomposed matrices. Second, the computational complexity of doing SVD on an m-by-n matrix is *O(min(mn2 , m2n))* [[26]](https://paperpile.com/c/CvqbpE/HffSf). Therefore, it is costly to run SVD on a large matrix and have to update the predictions every time there are new users and products entering the system. However, there are some heuristics that would allow us to update decomposed matrices with new information or perform matrix decomposition such as the incremental and folding-in techniques [[27,28]](https://paperpile.com/c/CvqbpE/VoEcy+TIPDS) and use a probabilistic model or gradient descent to decompose matrices [[29,30]](https://paperpile.com/c/CvqbpE/SXmod+wMryQ).

Because SVD can collapse related items into topics, it is often used in information retrieval to deal with the problem of synonymy (words with the same or similar meanings) and polysemy (words with multiple meanings). Using SVD, documents can be indexed with a smaller set of dimensions instead of all the words in the dictionary [[7]](https://paperpile.com/c/CvqbpE/oXPGi). This is a technique called Latent Semantic Indexing (LSI) or Latent Semantic Analysis (LSA) and it is the main idea of the retrieval method in this thesis.

### Latent Semantic Analysis (LSA)

LSA is a technique of using SVD in information retrieval, which is about obtaining relevant resources to information from a large collection of information source [[31]](https://paperpile.com/c/CvqbpE/0mZO7). This field began with text retrieval of library records and articles. It involves using a given query to find documents that are relevant to the query we want. For example, to get information about real estate in Vancouver, one might use the keyword query "Vancouver real estate", a statement of information needs. The text retrieval system use these keywords to find documents that are relevant to what the user would want.

The corpus of text is usually stored as a document-term matrix where the entries are either binary incidence or the total occurrences of the word in the document [[6]](https://paperpile.com/c/CvqbpE/Y49Aw). With this data structure, we could treat the query as a term vector and return all the documents whose term vectors are closest to the query vector [[7]](https://paperpile.com/c/CvqbpE/oXPGi). In other words, each document in the corpus is treated as some unstructured text or a bag of words, in which the word order does not matter and that a document is defined by the keywords it has [[7]](https://paperpile.com/c/CvqbpE/oXPGi). Finding the relevant documents to the query is the same as treating that query as a document and finding ones whose word vectors are similar to it.

However, literal matching of the words works poorly in retrieving relevant information due to synonymy and polysemy of words [[7,31]](https://paperpile.com/c/CvqbpE/0mZO7+oXPGi). For example, some documents relevant to the query "Vancouver real estate" might use phrases like "housing price" or "living cost" instead of explicitly "real estate". By naively checking the keywords, the text retrieval would miss relevant documents that use the alternative phrases to refer to real estate.

Latent semantic analysis is a way to overcome such problem by surveying a large corpus and estimating the patterns in word usage [[6,31]](https://paperpile.com/c/CvqbpE/0mZO7+Y49Aw). As described earlier in the example of a user-item matrix, the left decomposed matrix U from factorizing a document-term matrix is a document-concept matrix. SVD has the advantage of dealing with multiple problems associated with other text retrieval methods: synonymy, polysemy, sparseness, noise, and term-independence [[31,32]](https://paperpile.com/c/CvqbpE/uj0fY+0mZO7).

Synonymy would make documents that are highly related to seem dissimilar if they use a different set of synonyms. In the case of polysemy, word vectors from dissimilar documents would seem close. Sparseness refers to the many zeros in the document-term matrix because words contain in any document is much fewer than all the possible words. Word interdependence refers to the probability of observing a particular word is not independent of other words. Therefore, we cannot rely faithfully on Naive Bayes rule to use word probability to aid us in classifying documents or searching for related articles. Because SVD can pick out structures of word usage from the context of the entire document corpus, the rank-reduced factorization gives us the dimensions that preserve the main relationship between words and discard the noise that does not fit into the patterns supported by the entire corpus [[6,31]](https://paperpile.com/c/CvqbpE/0mZO7+Y49Aw).

Driven by an increasing number of publications that contain computational and experimental biomedical data in the field of genomics and proteomics, there has been much interest in leveraging literature-mining and machine learning techniques to help sort through the data [[33,34]](https://paperpile.com/c/CvqbpE/qmGxs+JZTOL). Therefore, many have applied LSA to discover knowledge and link information, including constructing protein-protein interaction networks, annotating gene names, and finding previously unknown gene-disease relationships [[2,34]](https://paperpile.com/c/CvqbpE/JZTOL+ScRJO). They would use a term co-occurrence matrix to find the underlying relationship between terms using the decomposed matrices and the new associations in the reconstructed matrix [[2,33]](https://paperpile.com/c/CvqbpE/ScRJO+qmGxs). Using LSA to find term-term relationships has been explored but not with the objective of finding document relationships. The objective of this thesis is to test whether LSA would be a good method to find similar articles.

To use LSA to retrieve similar biomedical documents, there are several implementation considerations including the keywords to use, the method of recording term occurrence in the matrix, numbers of ranks, similarity function to measure the distance between two concept vectors. Just like the case of many recommendation system applications, the decision on different parameters, similarity functions, and other optimizations all depend on the data and user experience we want to achieve [[13]](https://paperpile.com/c/CvqbpE/KsnSi).

Biomedical papers faced similar issues as other text retrieval applications such as synonym, and term-interdependency, as listed previously. For example, a paper can contain many synonyms, such as "tumour", "neoplasm", and “cancer".  Moreover, terms are often interdependent. For example, papers with the term "cancer" are likely to also mention terms such as "breast", "ovarian", and more. This demonstrates that LDA can be applied in information retrieval with biomedical literature.

Some text retrieval applications that use LDA consider every word in each document. In this study, we will instead use the Unified Medical Language System (UMLS), a repository of biomedical terms developed by the National Library of Medicine (NLH). The UMLS metathesaurus contains not just the MeSH terms, but also other knowledge databases such as Gene Ontology and NCBI taxonomy [[35]](https://paperpile.com/c/CvqbpE/ds0q4). By considering only the terms in UMLS, we can focus on biomedical concepts mentioned in the papers and limit the number of all possible terms so that our term-document matrix is more manageable.

Since we want to work with as many papers as possible and that abstracts are more available than full text, the corpus we will be using are the title and abstract of the paper. Second, there are several ways to record a term occurrence in a document. It could be a binary occurrence, term frequency (TF), or term-frequency and inverse-document-frequency (TF-IDF). Binary occurrence simply records whether a term exists in a document or not. TF assumes that highly occurring term in a document contributes more to the topic of the document. However, there are commonly used terms like "gene" and "DNA" that are not indicative of a biomedical paper's content. TF-IDF leverages both TF and term specificity to give more weight to terms that are better at discriminating different topics. Considering that an abstract is relatively short in comparison to a full paper, we cannot be certain of which measure would work best. We also test two different distance metrics, Euclidean distance and Cosine similarity, to see which one performs better at identifying similar documents.

While the best combination of the parameters is unknown, the quality of outcome also depends on the way we measure our success. The evaluation metrics would also need to provide us insights to improve the system, and how it relates to the user experience, data, and purpose.

### PubMed and other article recommenders

Since we want to determine how to retrieve similar biomedical articles, it is worthwhile to consider other recommendation systems for biomedical papers especially the ones that can retrieve similar articles. The most commonly used one is perhaps PubMed, the search engine for PubMed database that contains over 25 million references [[36]](https://paperpile.com/c/CvqbpE/WMOwR).

Recommendation systems that exist in the e-commerce world help people navigate through numerous products and find the ones they need. There is a similar requirement in the scientific community. New technologies continue to accelerate the pace of biomedical research, leading to an increasing number of scientific articles published every year. The question now is how technology can help a researcher concentrate her search on a few highly relevant articles to her work in lieu of reading everything.

There are several retrieval models, including vector models, probabilistic models, language models, and others including the modifications or hybrids of these three [[37]](https://paperpile.com/c/CvqbpE/yRxMv). We will focus primarily on the vector model and probabilistic model, which are part of the PubMed text retrieval method.

The probabilistic models began with the work of Maron and Kuhn [[38]](https://paperpile.com/c/CvqbpE/DFBH1). The approach requires each document to be indexed by a set of terms. Each term has a probability of how likely this term can be used to retrieve the article. There was also a significant effort toward computerizing text retrieval system. Another group led by Gerard Salton from Cornell University created a retrieval system named SMART and tested various different approaches of retrieval [[39]](https://paperpile.com/c/CvqbpE/gq5ih). One of them was TF-IDF[[40]](https://paperpile.com/c/CvqbpE/GlfCH), which combine two kinds of information. TF (term frequency), or the number of occurrences of a term in a document, indicates the general relevance of the term to the document. IDF, or inverse-document-frequency, is a factor that decreases as the total occurrences of a term in all the documents. The less frequent the word is used throughout the entire corpus the more informative the term would be [[40]](https://paperpile.com/c/CvqbpE/GlfCH).

The vector model represents each document d as a vector of terms that are each weighted by a global weight (*gwt*) and a local weight (*lwtd*):

|  |  |
| --- | --- |
|  | Equation 1.1 |

where T is the set of all key terms in consideration. By abstracting each document as a vector, we can measure the similarity between two documents by measuring the distance between them [[37]](https://paperpile.com/c/CvqbpE/yRxMv). This is analogous to our previous example in Figure 1.2. TF-IDF is a weighting scheme commonly used in a vector model.

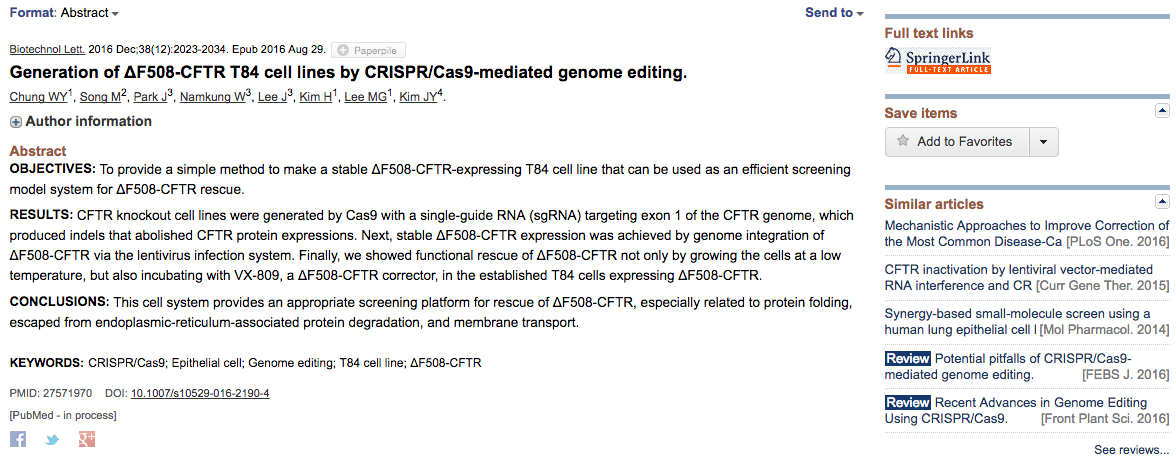
PubMed uses a vector model with a probabilistic approach [[41]](https://paperpile.com/c/CvqbpE/Aw6nU). Its search engine is perhaps the most well-known search engine for related biomedical articles. It retrieves related documents that researchers may also want to examine given the paper they are looking at (Figure 1.7). 

Figure 1.7 PubMed - similar articles

PubMed has adopted the traditional IDF to compute a global weight for each term except by taking the square root of IDF [[37]](https://paperpile.com/c/CvqbpE/yRxMv):

|  |  |
| --- | --- |
|  | Equation 1.2 |

where N is the total number of documents, and *nt* is the number of documents that contains the term t. The log is for smoothing the effect of common words. PubMed's the local weight is a modification of Harter's idea that the occurrences of unimportant words and important words follow different Poisson distributions [[42]](https://paperpile.com/c/CvqbpE/4ayxx). Given two rate constants *λi* and *λu* for important and unimportant words in a document, respectively, the local weight of the word in a document can be modelled as the probability that a word is important:

|  |  |
| --- | --- |
|  | Equation 1.3 |

where *dlen* is the length of Document *d*. The values of the constants were obtained by using the TREC 2005 data [[43]](https://paperpile.com/c/CvqbpE/TwrOc), a collection of human annotated abstracts, to do an exhaustive search for the optimal parameters. The evaluation method was based on the top five retrieved abstracts [[41]](https://paperpile.com/c/CvqbpE/Aw6nU).

Using this method, PubMed can then identify related articles for each record by performing pairwise analysis of shared terms between every record [[41]](https://paperpile.com/c/CvqbpE/Aw6nU):

|  |  |
| --- | --- |
|  | Equation 1.4 |

There are caveats to this approach. First, the keywords come from the Medical Subject Headings (MeSH), the National Library of Medicine's (NLM) controlled vocabulary thesaurus, and are manually assigned by indexers at the NLM [[44]](https://paperpile.com/c/CvqbpE/BWJBb). In the MEDLINE database, each paper has on average about a dozen MeSH headings to help characterize the content discussed in the article [[44]](https://paperpile.com/c/CvqbpE/BWJBb). MEDLINE is a database searchable under PubMed. While PubMed performs well in retrieving related documents, the MeSH terms involve a significant human effort to assign and might not be consistent among different indexers. Second, the vector model assumes that terms occur independently [[41]](https://paperpile.com/c/CvqbpE/Aw6nU), which is not true in biomedical literature.

Some have developed article recommenders using different heuristics. For example, Scienstein uses citation analysis and author analysis to find similar articles [[45]](https://paperpile.com/c/CvqbpE/fnCVk). CiteSeer uses a co-citation analysis and paper metadata in their calculation [[46]](https://paperpile.com/c/CvqbpE/0WnKY). Lee et al.'s recommender assumes that people are interested in the papers that are relevant to the ones they have written [[47]](https://paperpile.com/c/CvqbpE/oW61J).  Sujiyama et al. developed an article recommender system that uses researchers' publications as their preference profiles [[48]](https://paperpile.com/c/CvqbpE/4yor7). QxMD [[49]](https://paperpile.com/c/CvqbpE/DrMLt), a mobile application, notify users of papers published by journals that the users have followed. These methods presented valuable ways to construct a recommender system that would fit some needs. However, they often require information that is not always available, such as full text, MeSH keywords, citations, and past publications. In this thesis, we propose using dimensionality reduction, a technique closely related to collaborative filtering, to find related articles with only the abstract and title text in the papers.

In summary, there are many published papers for any given biomedical knowledge domain. Considering that the number of articles is measured in millions and continually growing, there are likely few topics in biomedicine in which one could hope to read all the available papers. While it is unlikely that any method would completely remove the risk of missing important information [[37]](https://paperpile.com/c/CvqbpE/yRxMv), we do need to rely on algorithms to help us find the articles that are most useful to us. Many modern recommenders in e-commerce use dimensionality reduction technique to help people to find relevant products and to explore other options that they might not have considered. In the scope of this thesis, we will look at how well latent semantic analysis, a dimensionality reduction application, can retrieve relevant paper for a given article.

## Parameters for retrieving relevant abstracts

### Overview

Going from collecting abstracts to making predictions for relevant articles involve several steps. Each step also can take on different parameters. For example, there are various methods of weighting term occurrences in documents and different metrics of calculating the similarity between documents, as shown in Table 2.1.

|  |  |  |
| --- | --- | --- |
| **Purpose** | **Data** | **Parameters** |
| Test different term weighting schemes for the terms in each abstract document. | Matrix, input to SVD | Term-frequency (TF), Binary, Term-Frequency and Inverse-Document-Frequency (TF-IDF) |
| Finding the metric of measuring the distance between two vectors that is the best for the document-semantic vectors | Decomposed matrix (document - semantic matrix) | Euclidean distance  Cosine similarity |

Table 2.1 Parameters for document retrieval

The parameters that are most effective at retrieving similar articles within small corpora in this experiment are then verified with larger matrices and used in the later experiments. The number retrieved PubMed’s related articles determined the precision value.

### Methods and tools

Going from the raw text to predictions of relevant articles involves several steps. Figure 2.1 Data and tools for testing different parameters on retrieving biomedical abstracts summarizes the workflow of this experiment. The installation scripts of the tools and download scripts of biomedical text and UMLS word list are in the GitHub repository for this project [[50]](https://paperpile.com/c/CvqbpE/CHnUe).

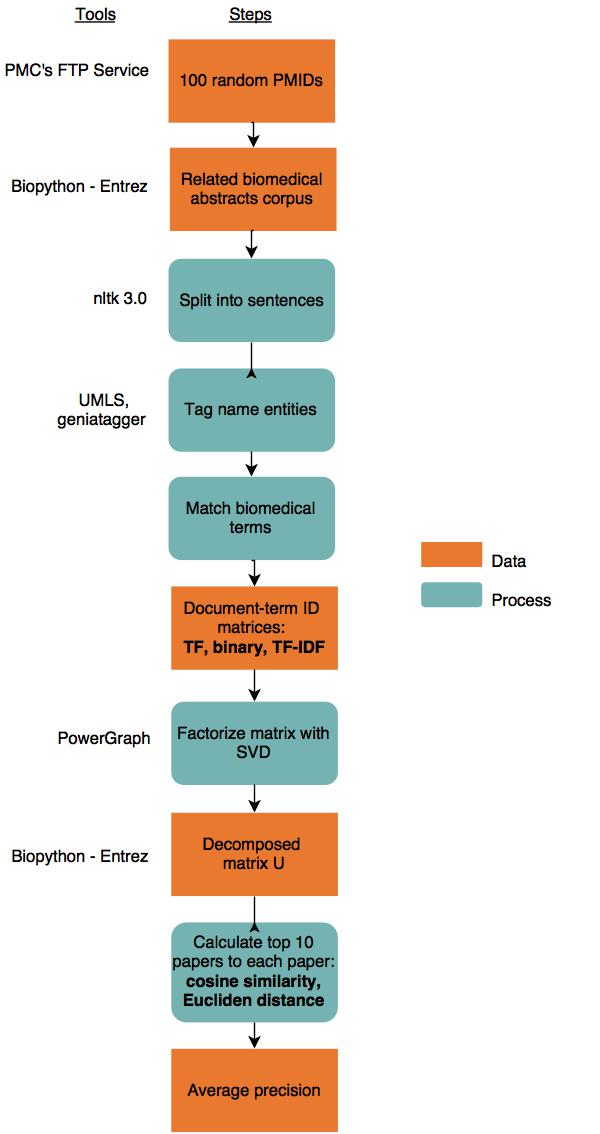


Figure 2.1 Data and tools for testing different parameters on retrieving biomedical abstracts

#### Obtaining biomedical abstracts

We prepared a collection of all available biomedical abstracts that will be used in the later experiments when we test whether we can retrieve similar articles from a corpus of millions. This collection was obtained through MEDLINE, the National Library of Medicine (NLM) journal citation database, which currently contains approximately 22 million citations [[51]](https://paperpile.com/c/CvqbpE/7Lr26).  Its citation records are available for download via bulk download [[36]](https://paperpile.com/c/CvqbpE/WMOwR). The citation records contain title, abstract, authors, and other metadata. Some records such as conference talks or biographies lack abstracts. We did a bulk download in January of 2015 and obtained about 13 million citations that contained abstract text.

To create a corpus that contains articles similar to each other, we used Entrez to retrieve articles that are related. Entrez has a Python package that is part of BioPython [[52]](https://paperpile.com/c/CvqbpE/TFeWE). It contains eight E-utilities which allow access to 48 different National Center for Biotechnology Information (NCBI) databases covering a variety of biomedical data such as gene records, protein sequences, and biomedical literature [[53]](https://paperpile.com/c/CvqbpE/say7b). The utilities use fixed URL syntax and input parameters to retrieve the requested data. Using the E-link utility on the PubMed database, we can obtain a list of PMIDs of the articles that PubMed's system consider related to the paper of a given PMID. We can then use the E-summary utility to get the corresponding abstracts for these PMIDs.

For each abstract corpus, we randomly selected 100 PubMed ID (PMID) from the list of 13 million citations, using the “shuf” command in bash. Using each of the PMID, we used Entrez.Bio to obtain a list of ranked relevant documents from PubMed. We repeated this process 100 times to make 100 abstract corpora and discarded incomplete ones that have missing abstracts due to failed API calls. We ended up with 92 such document corpora. Each collection has about 12,000 abstracts since PubMed has computed about 100 related articles for each of its record. Figure 2.2 shows the steps of constructing abstract corpora.

[https://lh5.googleusercontent.com/Wn_TXKmfNIRqkiDrNW_FB8Q3OFTheC5Ad7Yt9hDBmJ8EKdcVQYqTCJi1MqRmnG3Ocam5HLoxVRm2Wt45gnd_YgKSUOun_ZC9t6l5DysijvnCUPYy5xUJREcRxE_LtZ7a7-P_ohp_](https://drive.draw.io/#G0B2wyfz9_OACWd3NRSUx2NVhDbXc)

Figure 2.2 Making an abstract corpus

#### Text processing

The next step was to build a document-term matrix. To do so, we followed the bag-of-words model. We then identified all biomedical terms in each document such that terms such as "lung cancer" would be treated as one term instead of two separate words.

We used NLTK (v3.0.3), a Python library for many natural language processing tasks [[54]](https://paperpile.com/c/CvqbpE/SJBqP), to split each abstract into sentences. GENIA Tagger (v3.0.1) then extracted biomedical terms from each sentence. GENIA Tagger is a part-of-speech tagger that was trained on newspaper and biomedical text corpora such that it has high precisions in tagging biomedical terms [[55]](https://paperpile.com/c/CvqbpE/bZfQa). Splitting sentences prior to using GENIA Tagger ensures that the tagging process performs better.

The tag terms were then matched to a word list made from UMLS to obtain their corresponding term ID. The word list has taken different spellings, stemming, and synonyms into account. For example, "brain concussion", "brain concussions", "cerebral concussion", and "encephalopathy" would be assigned with the same term ID. Drug names such as "cl-1848c" and "cl 1848c" were also mapped to the same ID. The UMLS word list contained about 1 million biomedical term IDs, each of which mapped to one or more biomedical terms of the same meaning. In our UMLS word list file, each group of terms were written in one single line. We conveniently used the line numbers of the term groups as their term IDs.

The occurrences for the terms in each abstract were also recorded along with their IDs. In other words, we used the bag-of-word model commonly employed in many natural language processing and text retrieval applications. In this model, the text is treated as a bag of its words without any information of the relationships among the terms, the order in which they appear in the text, and the grammar in the sentences. This model allows us to record a document as a term vector, a simplified representation of the document. With multiple term vectors, we can then represent the entire corpus as a document-term matrix shown in Figure 2.3. Each row in the matrix is a word-vector for an abstract. Each column index is the ID for the term. This representation is useful in natural language processing.

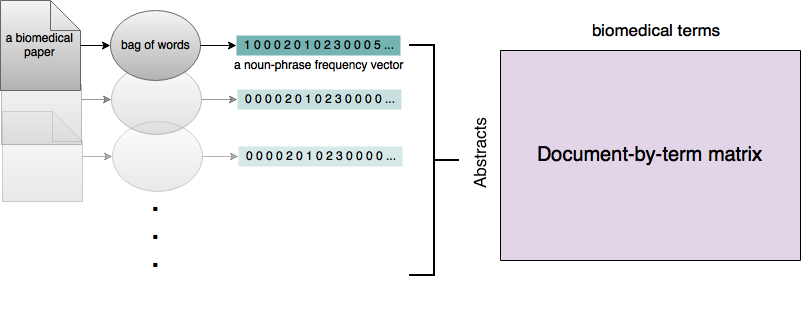
[](https://drive.draw.io/#G0B2wyfz9_OACWOVJMQ3I2VTdaa3c)

Figure 2.3 Bag of words model for biomedical literature

The entries inside document-term matrix describe the occurrences of the terms or the term frequency (TF) in each document. Besides TF, there are also other weighting schemes that determine what value each entry in the matrix would take. Some common weighting schemes include binary occurrences and TF-IDF. We will compare them against the document-term matrix that contains raw counts of the terms. We called the first matrix a term-frequency matrix.

Replacing all the non-zero entries with 1s creates a binary term-frequency matrix from a term frequency matrix. By doing so, we assume that relevancy of a document does not increase with the term frequency and that simply a Boolean for the term incidence in an abstract is sufficient.

TF-IDF weighting scheme takes a different approach. It assumes that the relevance of a term to a document does increase with its counts and a rare term is more informative about the topic of a document than frequently occurring terms.

|  |  |
| --- | --- |
|  | **Equation 2.1** |
|  | Equation 2.2 |
|  | Equation 2.3 |

In these equations, *N* is the number of total documents in a corpus, *d* is a given document, *TFd,t* is the number of term occurrences for term *t* in document *d*, and *nt* is the number of documents that contain term *t*. There are variants of TF and IDF weights. We chose the most commonly used version in which the *TF* part lessens the effect of relevancy by frequently used terms in a document. The log normalization reduces the effects of keyword stuffing, a practice of loading a document with keywords to increase its relevance to a topic, and long abstracts that might contain a greater number of relevant terms by chance. The IDF weight the terms based on how well they can discriminate articles of different topics.

For each of the 92 corpora, we created three versions of document-term matrix: TF, binary, and TF-IDF. We then evaluated these 276 matrices based on how well their decomposed matrices can help us retrieve documents that are deemed relevant by PubMed's system.

#### Singular value decomposition and PowerGraph (GraphLab v2)

The singular value decomposition (SVD) of the document-by-term matrices was done through PowerGraph. While Python's numpy library can only perform SVD on small matrices, PowerGraph has been tested to factorize matrices with up to 3.5 billion non-zero entries [[52]](https://paperpile.com/c/CvqbpE/TFeWE).

PowerGraph is a toolkit of several machine-learning algorithms. GraphLab was first developed by a group at Carnegie Mellon University in 2009 and has evolved into PowerGraph, or GraphLab 2.0. It addresses the difficulties with natural graphs using an algorithm that can divide the graphs across different machines to allow efficient, iterative computations. Because of this abstraction and parallel computation, this C++ framework can run computation on a graph of million or billions of vertices and edges faster than other competing systems by some magnitude.

PowerGraph performs better than other graph-based distributed parallel systems in that it improves the partition of the graphs, especially those with high-degree vertices. It uses the GAS abstraction consisting of three phases: Gather, Apply, and Scatter. The Gather phase is similar to the reduce phase in MapReduce. It computes the information about a small neighbourhood in a graph, such as summing the values from vertices that are close to each other. The Apply phase updates the state of the vertex from the Gather phase. The Scatter phase updates the adjacent edges and vertices and usually triggers future computations. This process repeats until convergence [[56]](https://paperpile.com/c/CvqbpE/FwWz7).

This abstraction is common across many implementations of machine learning algorithms. The major bottleneck of the computation comes from the Scatter phase, where the machines holding different parts of the graphs pass the information to one another. Graphs are commonly split by randomly assigning vertices to different machines, and this results in many cuts in the edges and more communication between machines. PowerGraph splits the vertices and assigning edges to different machines with some heuristics to minimize the communication between machines. This results in an improved performance on real-world graphs, as demonstrated through empirical evaluation [[56]](https://paperpile.com/c/CvqbpE/FwWz7).

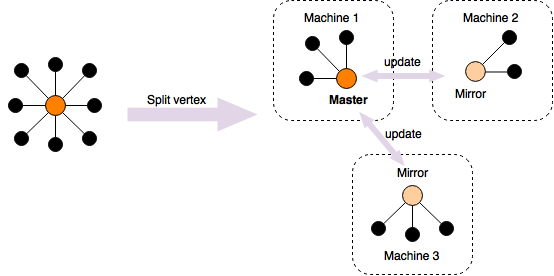
[](https://drive.draw.io/#G0B2wyfz9_OACWUU5ZOTNmOEJTdlE)

Figure 2.4 PowerGraph's distributed parallel computation

As shown in Figure 2.4, vertices, rather than edges, are split across different machines in PowerGraph. One vertex is designated as "master", which then collects computation from its mirror vertices in the Gather phase. In “Apply”, the information from the master node is passed to its mirrors. Then the Scatter phase triggers more computation on the individual machine.

PowerGraph contains many algorithms in machine learning, such as collaborative filtering and computer vision. The iterative algorithm for solving SVD in PowerGraph was implemented with the restarted Lanczos algorithm [[57]](https://paperpile.com/c/CvqbpE/K2hQO). In brief, Lanczos is an iterative algorithm with several matrix-vector multiplication steps to approximate each eigenvalue. PowerGraph uses this algorithm to approximate a specified number of singular values for an input matrix. It also provides an error estimate for each singular value.

The singular values and the errors from PowerGraph provide an estimate on how well the reduced SVD approximates the decomposed matrices. The error estimates start to increase as the singular values of higher dimensions become smaller and smaller. After discarding those singular values, we can then use the first few with lower errors to calculate the "coverage" in terms of singular values and the Frobenius norm.

The Frobenius norm of an m-by-n matrix is defined as the square root of the sum of the absolute squares of its elements, as shown in the following equation [[58]](https://paperpile.com/c/CvqbpE/om6qV).

|  |  |
| --- | --- |
|  | Equation 2.4 |

The Frobenius norm of a matrix is also equal to the square root of the sum of the absolute squares of its singular values from a full SVD. In a reduced SVD that uses only the first k singular values to do a k-rank approximation of the matrix A, the trailing singular values after kth are set to zeros, as shown below [[59]](https://paperpile.com/c/CvqbpE/YTvgs).

|  |  |
| --- | --- |
|  | Equation 2.5 |

We define coverage of a k-rank approximation as the ratio of the Frobenius norm of Âk to ||A||F.

In summary, we used the restarted Lanczos algorithm implemented in PowerGraph's collaborative filtering to do a low-rank approximation a matrix, or a k-reduced SVD, to obtain the decompose matrices. The error estimates tell us how many singular values are usable. The coverage, or the ratio between the two Frobenius norms of the approximate and original matrices, is a sanity check on how much information of the original matrix is captured in the decomposed matrices for a given *k*.

#### Calculate relevant documents

SVD in PowerGraph outputs the three decomposed matrices: the left decomposed matrix U, a list of singular values and their error estimates, and the right decomposed matrix V. The matrix *U* represents the document by concept matrix. The columns of U can be considered as a collection of terms contributing to a collective concept at different weights. The weights are the singular values. Each row is a concept vector for that document. Calculating the distance between any given two vectors is a measure of how far apart the two corresponding documents are from each other. The two metrics that we used to measure similarity of two vectors are the Euclidean distance and cosine similarity. Given two vectors p and q of n dimensions, these are calculated as followed:

|  |  |  |
| --- | --- | --- |
| *Euclidean distance* |  | Equation 2.6 |
| *Cosine similarity* |  | Equation 2.7 |

Euclidean distance is the length of a straight-line between two points calculated with the Pythagorean formula. It considers the documents as points in a space of n dimensions. Similar to RMSE (root-mean-square error), the square of the difference removes the negative values and exaggerates the bigger differences. We can also consider the two documents as vectors in the n-dimensional space by using cosine similarity to measure the angle between their vectors. The smaller the angle or the cosine of the angle, the more similar the two documents are.

Since each singular value captures the importance of each dimension, we first scaled the entries of the row vectors in *U*, the document by concept vectors, with the corresponding singular values before calculating their distances. We also vary the number of dimensions to see how they affect the result. We calculated the pairwise distances between our target abstracts to all other abstracts then sorted the abstracts by the distances to each targeted abstract. We then looked how many of the top ranking articles identified by our method agreed with PubMed's recommendations.

In summary, SVD gives us a document-by-concept matrix and a singular value for each concept dimension. We scaled document-by-concept matrix with the singular values and measured the difference between pairs of documents with Euclidean distance and cosine similarity. We also varied the number of dimensions to see how the performance changes as we represented the abstracts with different numbers of concepts.

#### Evaluation: precision and singular values analysis

In recommendation systems, evaluation methods need to fit with the expected usage. In many use cases, people typically do not look beyond the first few recommended items. Therefore, we used the top ten abstracts for each targeted abstract to evaluate our method. The **average precision** for our top 10 retrievals measures how relevant our top ranked abstracts are by PubMed’s standards.

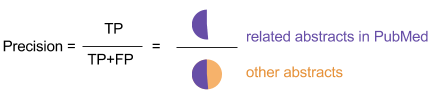


Figure 2.5 Precision of the retrievals by SVD

For each targeted abstract, the precision is measured by the ratio of the related abstracts in PubMed (TP: True positive) to the total number of retrieved abstracts. In our experiment, we are using a total of 10 abstracts. We considered the retrieved articles that are not in PubMed's list as false positives. For each corpus, the performance is measured by averaging the precisions for each of the 100 papers and their top ten relevant abstracts.

We calculate this average precision for each of the 92 abstract corpora at different number of singular values, as well as with different distance metrics (cosine similarity and Euclidean distance) and term-weighting schema (TF, binary, TF-IDF).

### Result

The overall result in this chapter is summarized in Figure 2.6. Each dot in the graph is an average precision of retrieving ten related papers for each of one hundred targeted abstracts in one corpus. The lines on the curve lie on the averages of the 92 different values at each combination of distance metric, weighting schema, and number of singular value (NSV).

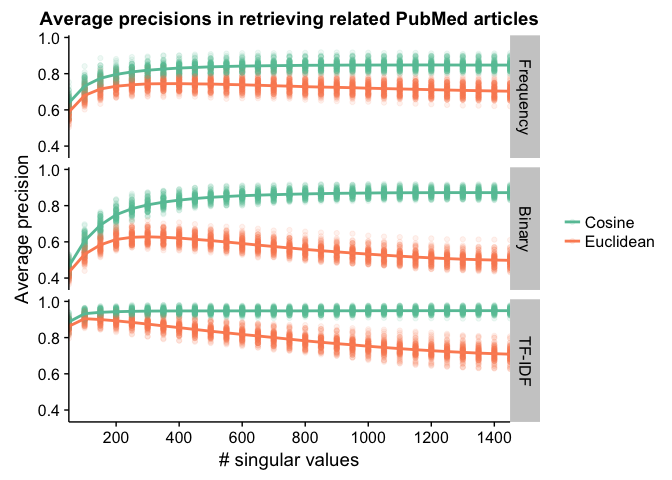


Figure 2.6 Average precision over top 10 abstracts for 92 different matrices

In all three matrices, using cosine as the similarity measurement led to better precision. Moreover, the TF-IDF matrices in combination with cosine similarity as the distance metric achieved better precision at a lower NSV (number of singular values) than other matrix-type and distance metric combinations. Some curves show that after some number of singular values, including more dimensions in SVD actually lowered the average precisions.

Figure 2.7 summarizes the maximum precisions for each of the 92 matrices with different combination of weighting schema and distance metric. It shows that using TF-IDF and cosine similarity has the highest maximum precision.

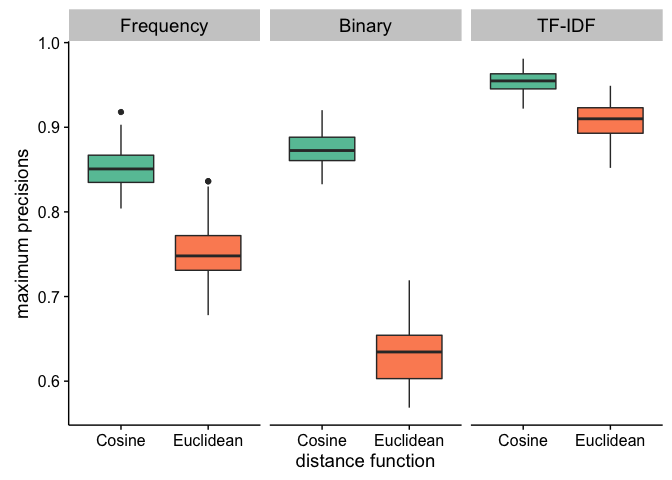


Figure 2.7 Maximum precisions for the 92 abstract corpora

Figure 2.8 and Figure 2.9 represent a sanity check on the singular values and their error estimates. The standard deviations for the averages for both plots also follow similar trends (not shown). Figure 2.8 shows that error estimates for the singular values in TF-IDF matrices increase earlier than the other two matrices, and that most of the singular values after the 1000th one had high error estimate and thus likely unreliable. Figure 2.9 shows that the first few singular values in the term frequency matrices are larger than the other matrices.

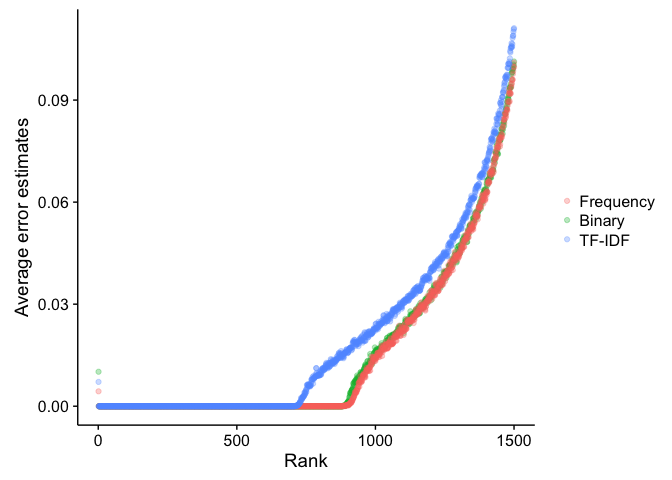


Figure 2.8 Average error estimates from 92 abstract corpora

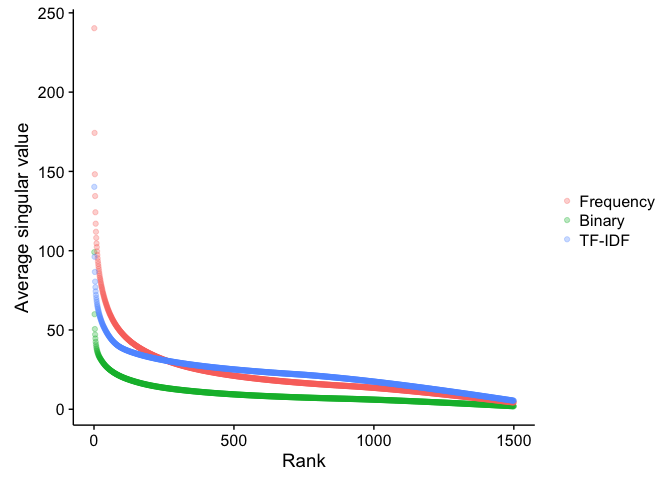


Figure 2.9 Average singular values from the 92 abstract corpora

We also looked at the coverage achieved at different number of dimensions by the three different types of matrices. Figure 2.10 indicates that both binary matrices and term-frequency matrices can achieve relatively high coverage with less number of dimensions comparing to TF-IDF matrices. For example, the first 300 singular values in a binary matrix can achieve about 80% of coverage, whereas in TF-IDF the coverage with 300 singular values is less than 60%.

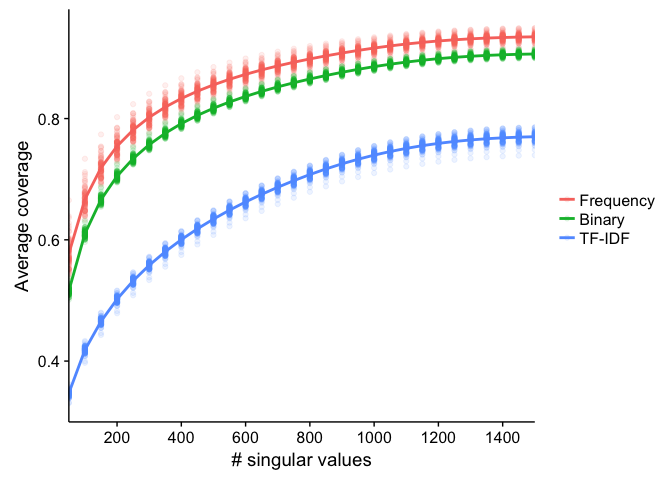


Figure 2.10 Average coverage of the 92 abstract corpora

Figure 2.11 summarizes the average running time at different NSV to compute for top 10 relevant papers for each of the 100 target papers in each abstract corpus, using the Euclidean distance and cosine similarity functions from SciPy Python library [[60]](https://paperpile.com/c/CvqbpE/D6fqV). It shows that calculating Euclidean distances is more computationally expensive than calculating for cosine similarity.

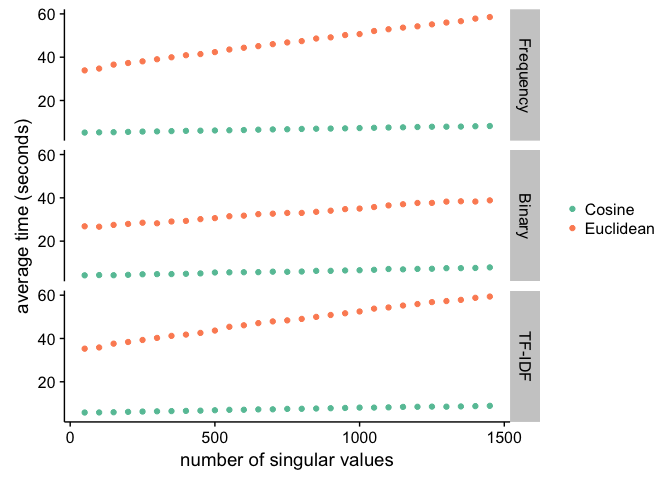


Figure 2.11 Running time with different distance metric

### Discussion

We found that regardless of the type of matrices, the cosine similarity function was better at retrieving relevant articles than the Euclidean distance function. It also took less time to compute the cosine similarity between two document vectors, as shown in Figure 2.11. Therefore, cosine similarity is the distance function chosen for future experiments. We determined that we would use this distance function when we later attempt to retrieve relevant documents amongst 13 million articles.

We also see that TF-IDF weighting scheme performed better than the other two. Using cosine similarity function to measure distances between vectors in a decomposed U matrix from a TF-IDF document-term matrix seems to work the best compared to other combinations, at least in the case of the small matrices that contained about 12,000 abstracts. While Frobenius norms calculated from the singular values of TF-IDF matrices achieved lower coverage, TF-IDF matrices had better results than the binary and TF matrices.

Our abstract corpora came from 100 randomly chosen papers and their related articles. If these articles were very different, each corpus would contain only 100 different topics. Therefore, we may not easily generalize this conclusion to say that TF-IDF would be the best choice for the big matrix that contains all 13 million abstracts. In the later chapters, another experiment was done to further determine which weighting metric is better.

Evaluation in this chapter used PubMed as the conceptual “gold standard”. We built our abstract corpora by retrieving relevant articles using Entrez, and calculated the precision by looking at whether we retrieved those relevant articles. Since PubMed's relevant articles metrics have been refined over many years and are based on human annotations, we assume that PubMed’s relevant articles were truly related to their targeted articles. This gave us a starting point in establishing what parameters to use.

## Sampling word occurrences

### Overview

As discussed in the first chapter, the collaborative filtering part of the GraphLab software requires the matrix file and other parameters such as the number of singular values (NSV) that it would estimate. Given this NSV, GraphLab will then calculate the decomposed matrices with the given number of dimensions and the singular values with their estimated errors.  The singular values that can be used to calculate the predictions are those with low error rates. Frobenius norm describes how much the singular values represent the original matrix.

In the previous experiment, NSV is set to 1500 to perform SVD on matrices with about 12,000 abstracts. This number was enough to obtain the singular values needed to achieve a coverage norm of about 80% on a matrix with 12,000 documents. For example, the term-frequency (TF) matrices in the previous chapter only need about 400 singular values to achieve 80% of the coverage, as shown in Figure 2.10.

To test whether using singular value decomposition can scale for the entire PubMed database, we would need to run SVD on a matrix that represents the term occurrences in 13 millions abstracts. However, the running time for GraphLab's collaborative filtering, or its SVD algorithm, significantly lengthens with the size of the matrix and the NSV.

Using GraphLab to perform SVD on a matrix, we would get some estimated singular values with low errors while the rest have high error rate or do not contribute to 80% of Frobenius norm. Calculating for more singular values than what is needed would be a waste of computing resources and time. On the other hand, we could specify a lower number of singular values to lessen the running time and only to find out later that the total estimated singular values are not sufficient to provide enough estimation. We would then need to rerun SVD with a higher number of singular values.

To achieve a desirable coverage, how can one make an informed choice of NSV for running an SVD on a large matrix? In preparation to perform singular value decomposition (SVD) on the matrix with 13 millions abstracts, we applied the concept of Graph Sparsification.

Graph Sparsification is about approximating a graph G with a sparser graph G' such that G' is faster to compute and can be used to make some non-trivial statements, such as the number of clusters and shortest path, about the original graph G [[61]](https://paperpile.com/c/CvqbpE/saXVC). One way to do this is by sampling the edges in the graph. We can also use this idea to sample the word occurrences in a document-by-term matrix to create an approximated matrix that can be more easily decomposed with SVD.

To sample term occurrences, we would generate a random number between 0 and 1 for each occurrence of word recorded in our matrix and keep only those that are below a chosen threshold between 0 and 1 in the approximated matrix. As demonstrated in Figure 3.1, if a word *wi* occurs in a document *d4* three times, we can generate three random numbers, 0.35, 0.41, and 0.73 for the three occurrences. Suppose that our sampling rate is 0.50, the entry in the matrix at row j and column i is reduced from 3 to 2 because we discarded the occurrence assigned to the random number 0.73. If the sampling rate is higher, more term occurrences are kept. This sampling method reduces the density of the matrix because the low occurrences in the matrix are likely to be reduced down to zeros.

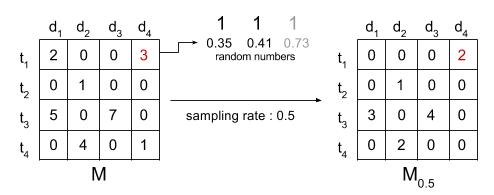


Figure 3.1 Random sampling on a matrix

Since the denser a matrix is, or the more non-zero entries a matrix has, the longer it takes for SVD to decompose the matrix. A lower sampling threshold produces a sparser matrix, which can be used as an input to SVD to estimate the number of singular value (NSV) needed to achieve a high coverage without spending a longer computing time on the original matrix to search for that NSV. Once this NSV is found, it can then be used for the SVD on the original matrix.

We can also think of matrix sampling as an indirect way of shortening our abstracts. By randomly taking out the term occurrences, we are essentially removing some sentences or parts of sentences from each abstract. Because each term occurrence has the same chance of remaining in the shorten abstract, more frequently occurred terms in the abstract would likely to be carry over to the sampled matrix. However, less frequently occurred terms that are potentially important to the topic of the abstract could also be taken away by a low sampling rate.

### Method

To test whether sampling matrix can help us predict NSV for a high coverage and how small the sampling rate can be, we performed sampling on ten different abstract corpora taken from the 92 abstract corpora in the previous experiment. We used the TF matrices of these ten corpora, and sampled their term occurrences at various rates for three times to ensure consistency. We then did SVD on these sparser matrices and looked at the coverage at different NSV.

To predict the NSV for a term-frequency and inverse-document-frequency (TF-IDF) matrix was less straightforward. The entries in a TF-IDF matrix are weights instead of whole numbers that can be interpreted as term-occurrences, as in the case of TF matrices. To sample a TF-IDF matrix, we tried two different methods. One method was first transforming its sampled TF matrix into a TF-IDF matrix and then looking at the coverage values at different NSV after doing SVD. The other method was scaling the TF-IDF matrices with a multiplier, rounding the values of the entries to whole numbers, and then sampling on the scaled TF-IDF matrices in the same way with TF matrices. We looked at which sampling method leads to more consistent coverage across different sampling rates.

Once we determined a reasonable sampling rate and an appropriate method for sampling TF-IDF matrices, we then used the method to predict the NSV for the TF-IDF and TF matrices constructed from the 13-million abstract corpus.

### Result

#### Term frequency matrix sampling

Since each of the TF matrices was sampled three times at nine different rates, we had 270 sampled matrices plus the ten original TF matrices. In Figure 3.2, each point is the averages of coverage measurements from the three sampling replicates of each matrix. The coverage rises as the sampling rate increases from 0.1 to 1 (no sampling) but is relatively flat for higher NSV. The numbers are identifiers for the ten abstract corpora.

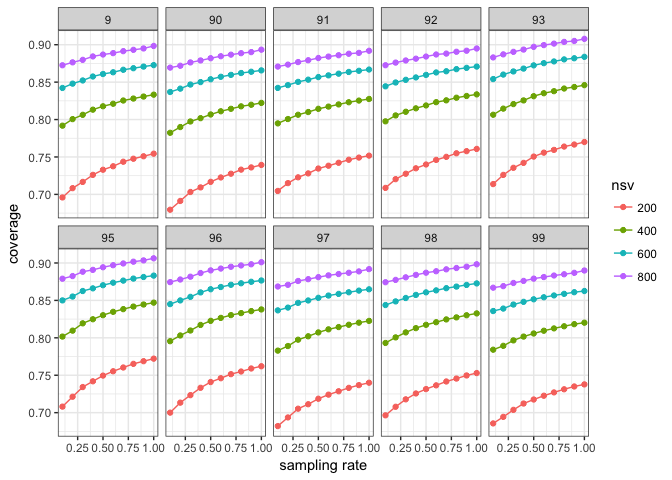


Figure 3.2 Coverage values at different NSV and sampling rate for 10 TF matrices

Figure 3.3 shows that, for sampled matrices that come from the same matrices, regardless of sampling rate, have similar error estimates for their singular values.

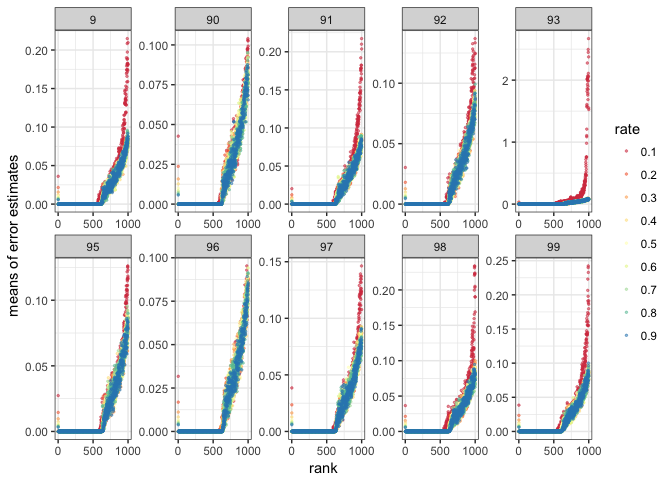


Figure 3.3 Error estimates of singular values for TF matrices

#### TF-IDF matrix sampling

The first attempt on sampling a TF-IDF matrix was doing sampling on the term occurrences and then converting the result into a TF-IDF matrix. Figure 3.4 shows the coverage by taking the TF-IDF forms of the sampled matrices from the previous section.

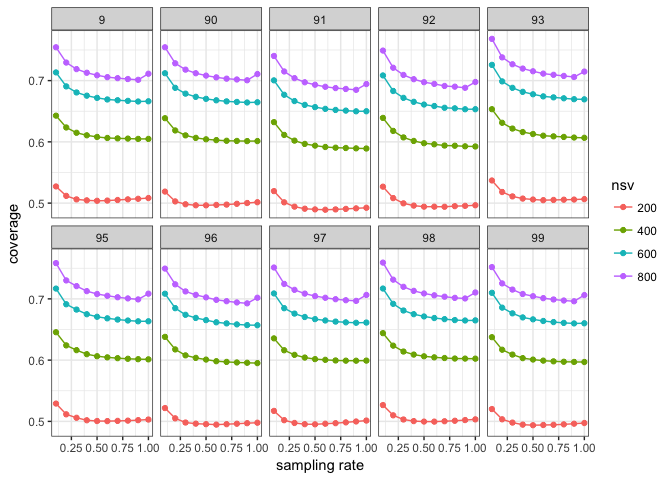


Figure 3.4 Coverage for TF-IDF forms of sampled matrices

Another attempt at sampling a TF-IDF matrix was by using a scalar to round up the float values into integers. We first converted the original term-frequency into TF-IDF matrices. These TF-IDF matrices were then scaled by three different factors (10, 100, 1000) and sampled at various rates (0.01, 0.05, 0.1, 0.2, 0.4) with three replicates. Since there were ten different original matrices, this experiment involved a total of 450 sampled matrices. We then did SVD on each of the matrices to see the effect of sampling on coverage at different NSV.

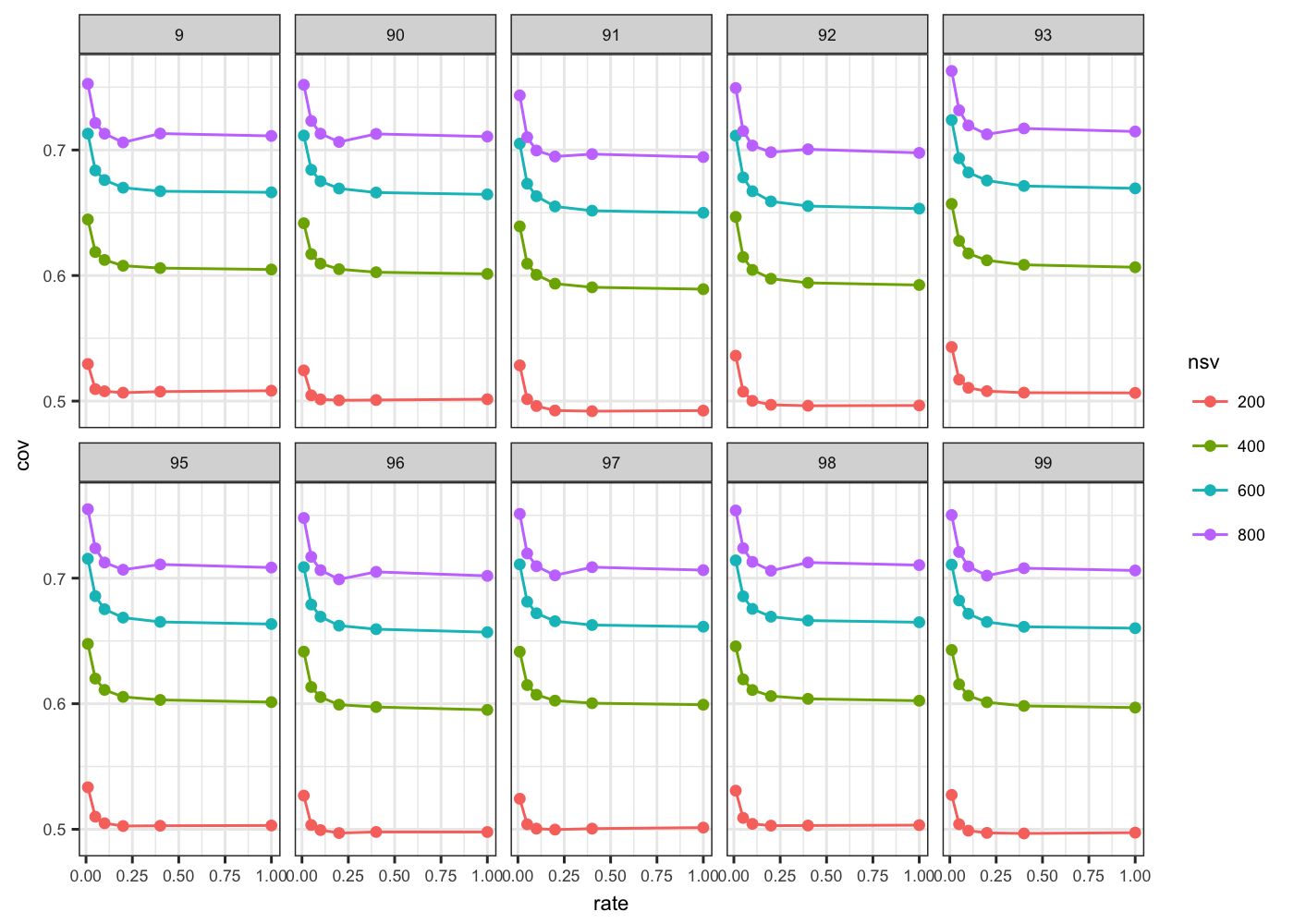


Figure 3.5 Coverage for TF-IDF matrix scaled by 10 and then sampled

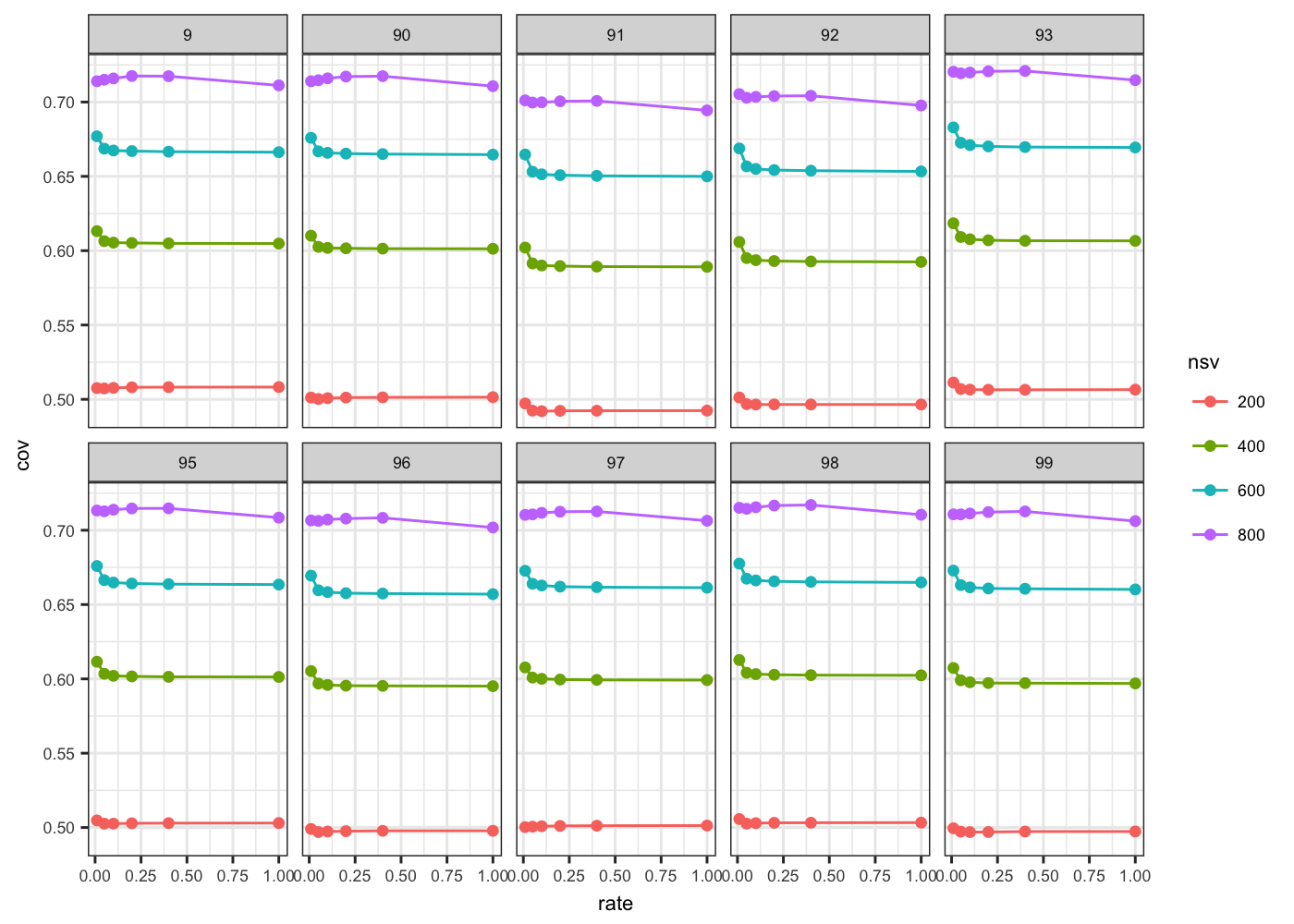


Figure 3.6 Coverage for TF-IDF matrices scaled by 100 and then sampled

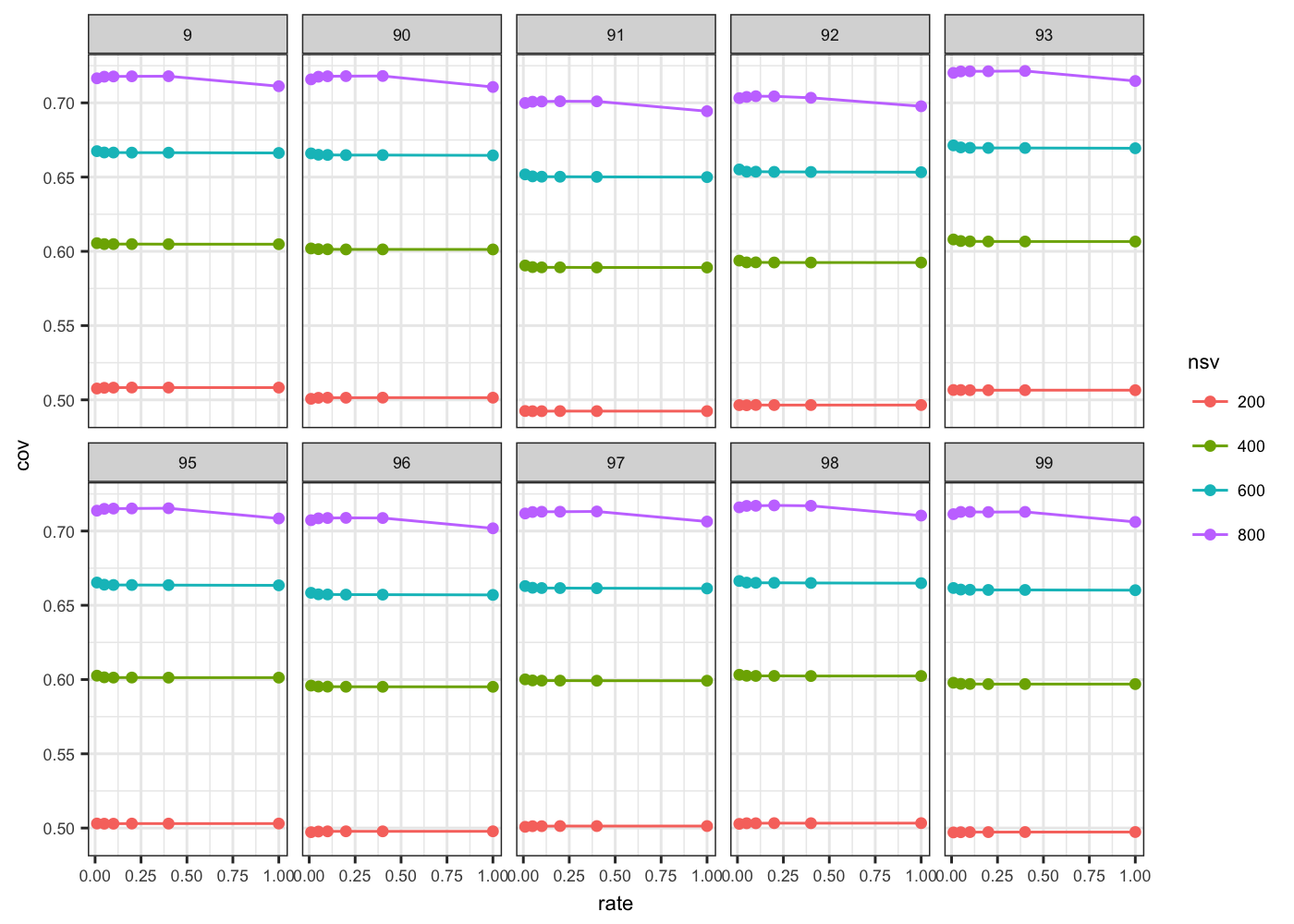


Figure 3.7 Coverage for TF-IDF matrix scaled by 1000 and then sampled

Figure 3.5, Figure 3.6 and Figure 3.7 show the coverage at different sampling rates and different NSV for TF-IDF matrices that were scaled by three different factors. The curves formed by the coverage from matrices scaled by a factor of 100 are somewhat flat, which shows that we can use a sampling rate of 0.01 on a TF-IDF matrix whose entries are scaled by 100 to approximate how many NSV we would need for the original TF-IDF matrix to achieve a similar coverage.

The next step was to run SVD on our 13-million abstract matrix. We sampled the TF matrix by 0.1 to predict the NSV we need for the original TF matrix and used the scaling method with the 100 scalar and 0.01 sampling rate to predict the NSV we need for the TF-IDF matrix.

#### Sampling on the 13 million abstract corpus

The TF matrix was sampled with the sampling rate of 0.1. We then did SVD on this sampled matrix with 2000 NSV. The error estimates show that around the first 1500 singular values were useful (Figure 3.10). From Figure 3.2, we knew that the coverage might be higher for the same NSV for the original matrix. Indeed, the coverage was higher at various NSV (Figure 3.8) when we estimated the first 2000 singular values. The singular values are greater in the full matrix due to the sampling but decay at around the same rank as in the sampled matrix (Figure 3.9).

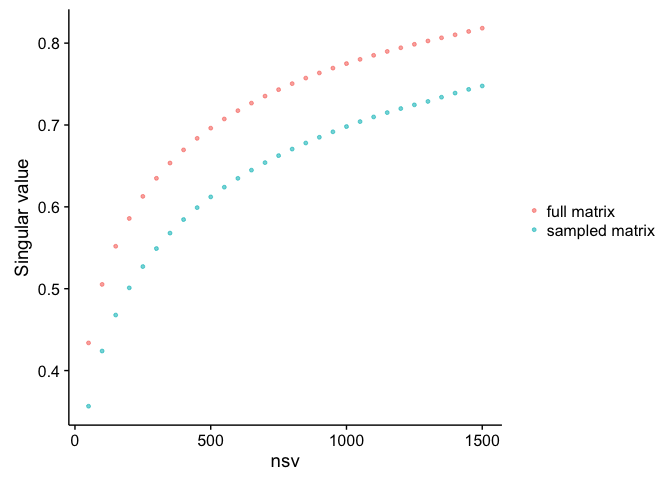


Figure 3.8 Coverage of the sampled and original 13M-abstract TF matrices

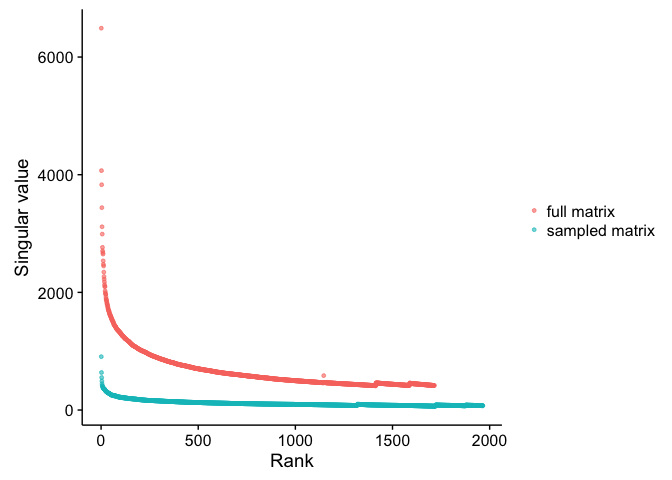


Figure 3.9 Singular values of the sampled and original 13M-abstract TF matrices

Figure 3.10 shows that only the first ~1000 singular values for the original matrix have low error estimates. Therefore, we will use only the first 1000 singular values for our TF matrix in the next experiment.

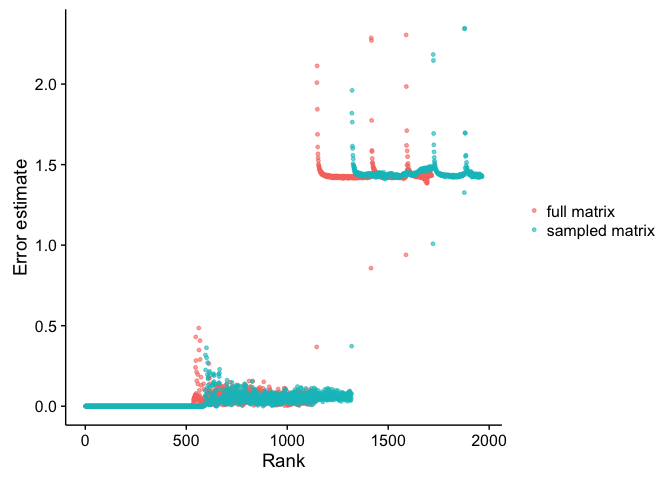


Figure 3.10 Error estimate of the sampled and original 13M-abstract TF matrices

The 13-million-abstract TF-IDF matrix was scaled by a factor of 100 and then sampled at a rate of 0.01. For estimating 2000 singular values, we see that only the first 1000 singular values might be usable (Figure 3.13). We tried to estimate also a similar number of singular values for the original TF-IDF matrix.

As expected, the coverage (Figure 3.11), values of singular values (Figure 3.12), and error estimate of the singular values (Figure 3.13) are similar for the sampled and original matrices. The sampled matrix is slightly sparser than the original matrix. Their ratio of number of entries is about 0.92. Therefore, this method of sampling a matrix to approximate a NSV needs a better combination of scaling factor and sampling rate.

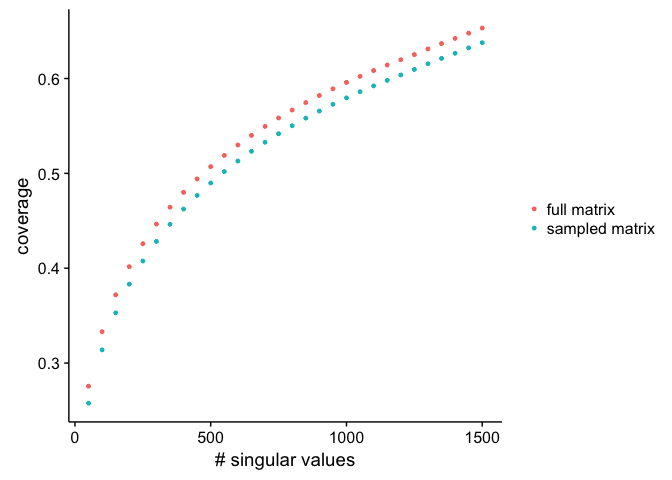


Figure 3.11 Coverage of the sampled and the original 13M-abstract TF-IDF matrices

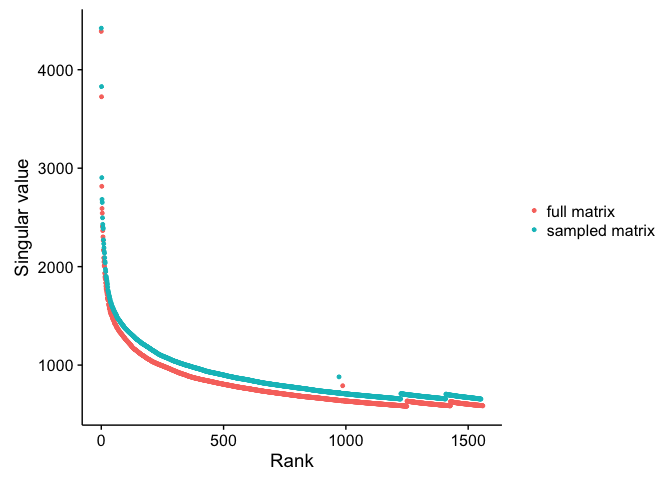


Figure 3.12 Singular values of the sampled and original 13M-abstract TF-IDF matrices

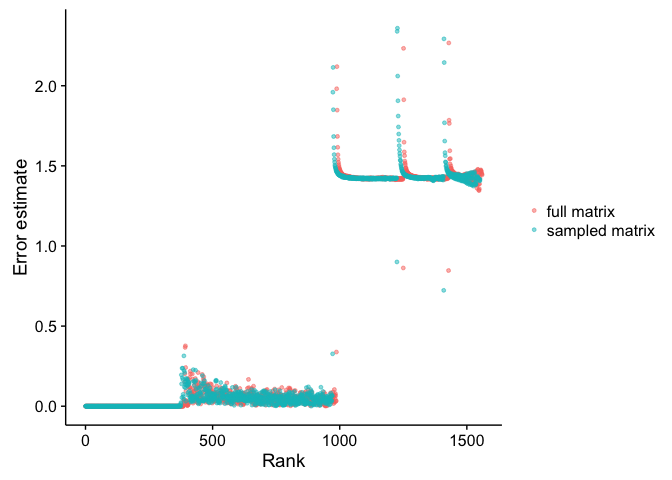


Figure 3.13 Error estimates of the sampled and original 13M-abstract TF-IDF matrices

### Discussion

This chapter focuses on how to make an educated guess on the NSV for a large matrix whose decomposed matrices would take days to compute with GraphLab's SVD. The idea is sampling the entries in the matrix to produce a sparser matrix whose decomposed matrices would be faster to compute. The result from the singular values of the sparser matrix can then be used to estimate how many singular values we need for the original matrix. For example, if 1000 singular values in the sampled matrix achieve a desirable coverage of 70 percent, then we would expect that we need about 1000 singular values to achieve a similar coverage on the original matrix as well. We tested the idea on ten TF small matrices by sampling their entries to produce their sparser counterparts and found that it was possible.

We tried two different methods of sampling a TF-IDF matrix. The "sample-first" method is by sampling the TF matrix and then converting the sampled matrix into its TF-IDF form. However, this method can conflict with the idea of TF-IDF, which gives higher weight to terms that are infrequent in the overall corpus. The less frequently occurring terms that are critical to the abstract's content can be filtered out during the sampling process. Therefore, the TF-IDF form of a sampled matrix might not be representative of the TF-IDF of the original matrix.

Another method was multiplying the values in the TF-IDF matrix with a scalar and round to the nearest integers, such that we can sample the integer values to produce a sparser TF-IDF matrix. We found that this method using 100 as the scalar worked better than the other method in producing consistent coverage at different sampling rates.

We applied this idea on our two large matrices, TF and TF-IDF matrices from the 13-million abstract corpus. The analysis on the singular values from decomposing the sampled and original matrices show that we could use sampling to help us run SVD with a reasonable NSV on a large matrix. The scaling and sampling of the TF-IDF matrix with 100 as the scalar and 0.01 as the sampling rate produced only a slightly sparser matrix due to the high scaling factor. Using the method of sampling first and then converting to the TF-IDF format might help improve the time needed to estimate the NSV. However, the matrix sampled in such way might not be representative of the original TF-IDF matrix.

## Evaluation with rankings and surveys

### Overview

In this chapter, we finalized our chosen parameter for using SVD to predict related papers. In chapter 2, we looked at using two different distance metrics: Euclidean distance and cosine similarity, as well as the three different weighting schemes for the term occurrences: term-frequency (TF), binary term-occurrence, and term-frequency and inverse-document-frequency (TF-IDF). We have chosen cosine similarity as our similarity metric because it performed consistently better with all three weighting schemes and took less time to compute. Other text retrieval systems also often used cosine similarity. However, we need additional validation to determine which weighting scheme to use for the large corpus with 13 million abstracts.

We examined how well each weighting scheme retrieved the relevant papers to some selected target papers. Once we determined the one where the relevant articles are ranked the highest, we then asked thirty-three students in the bioinformatics graduate program to be our testers who would evaluate how our method performs at retrieving relevant paper in comparison to PubMed through surveys. The survey result showed whether SVD with the parameters we have chosen could be a valuable alternative to manual keyword assignments in PubMed's text retrieval system.

### Method

#### Rank analysis

To determine how well each weighting scheme can rank the relevant papers highly, we randomly selected 10,000 articles from the 13-million document corpus as our targets. For each of these papers, we identified ten most relevant articles determined by PubMed.

The next step is to do SVD on the three different matrices: TF, binary, and TF-IDF. Using different number of singular values (NSV), we calculated the rankings of the top 10 papers for each 10,000 targeted abstracts using TrackRank [[62]](https://paperpile.com/c/CvqbpE/16SNn). For each targeted abstract, the program uses the absolute value of cosine similarity between each pair of vectors and records the rankings of those that are the top-10 most relevant papers in PubMed. At the same time, it records the closest paper to the targeted paper so that we have information about what is the closest article determined by our method.

We randomly selected 9,000 papers out of these 10,000 target papers and looked at the rankings of their 10 most relevant papers. We determined the best combination of weighting scheme and NSV as the one that have the most papers that fell within 10,000th ranking, or the top 0.07 percent of all possible 13 million rankings.

#### Survey analysis

We created surveys for our chosen testers to determine whether the most relevant paper from PubMed or the one from SVD is more related to a given target paper. We used the remaining 1000 target papers out of the previously selected 10,000 papers for our surveys.

First, a template survey is manually created in SurveyMonkey [[63]](https://paperpile.com/c/CvqbpE/2XQo1) to ensure the layout and the formats are optimal for viewing. Each survey contains ten pages of questions. Each question asks the respondents to read one target abstract and choose whether Abstract A or B is more related to the target abstract or that they are both equally irrelevant or relevant. These three choices are enough to determine the relative performance of SVD to PubMed, and vice versa. Figure 4.1 shows a question page in the template survey.

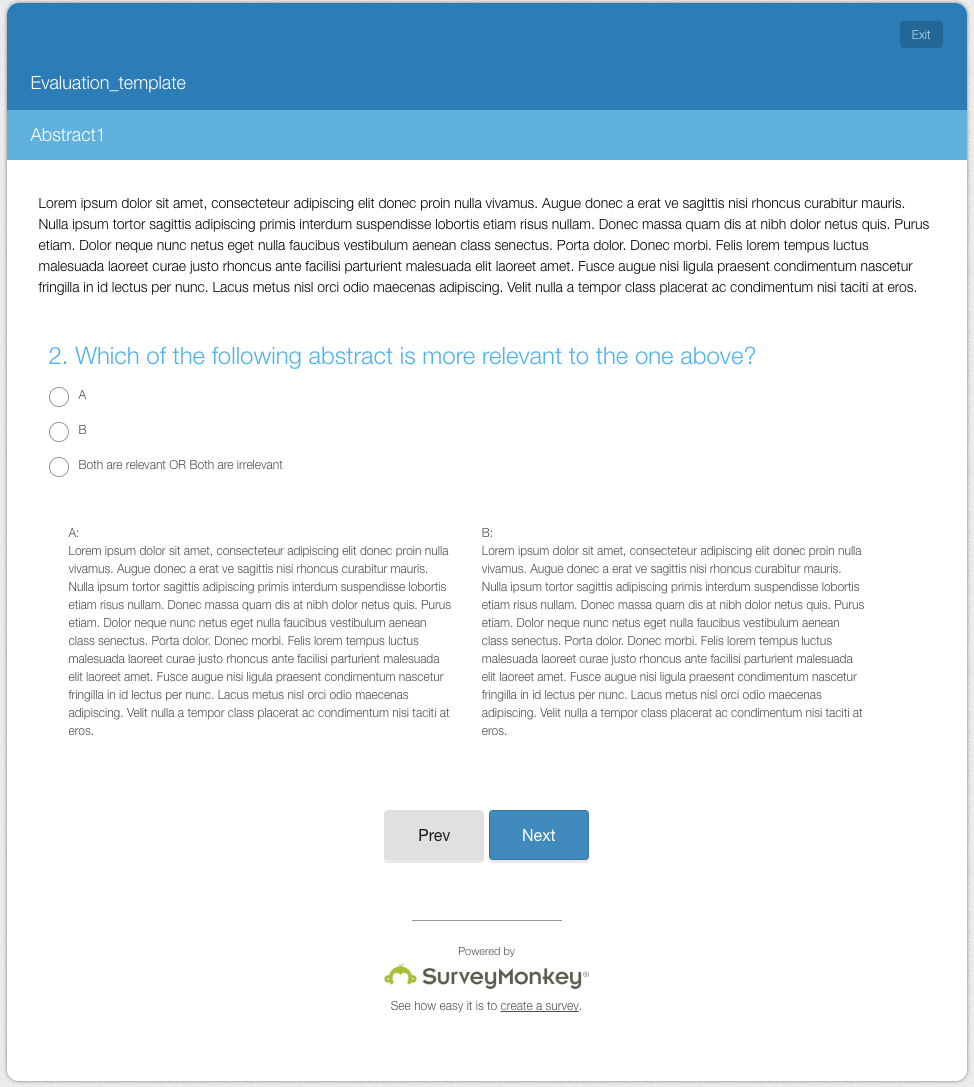


Figure 4.1 Question template for a survey

The 1000 target papers and their most related papers from SVD and PubMed were divided into 100 sets of 10 questions. Each set of 10 target papers and their related papers were used to create one survey. The most relevant abstracts from PubMed and SVD were randomly assigned to either Abstract A or Abstract B. Using the SurveyMonkey API [[64]](https://paperpile.com/c/CvqbpE/CcPHS), we automated the next few steps. For each set of 10 questions, we made a clone of the template survey and replaced the placeholder text with abstract and title text. We then created a web link to the survey for collecting the responses and retrieved the IDs for the choices for each question to identify the options that people chose for each question.

We randomly chose one survey for quality control, Survey 56, which was made available to all the respondents in order to gauge how consistently people judge the abstracts. The remaining 99 links to the 99 surveys were divided into 33 sets of three links. The link to Survey 56 is then put into each set. Every set is then shuffled so that the link to Survey 56 appeared in different order in the set of four links. We randomly chose 33 people from the Bioinformatics Graduate Program at UBC. We assigned each person to a set. Using the R package "gmailr" [[65]](https://paperpile.com/c/CvqbpE/IxKDu), an interface to the Gmail RESTful API, to create emails with the unique set of links for different people and automate the email sending process. Using the SurveyMonkey API and gmailr ensured that each step was done efficiently and accurately.

We gave people who received the survey links the following instructions. First, they would read the target abstract, Abstract A, and Abstract B. They would then judge whether Abstract A or B is more relevant to the target abstract and select the corresponding choice. In the event when both Abstract A and Abstract B are irrelevant to the target abstract, equally relevant to the target abstract, or identical, they would select third choice ("Both are relevant OR both are irrelevant"). If they did not feel that they sufficiently understand some of the abstracts in any question, they could simply skip to the next question without selecting a choice.

The results from all the 100 surveys were retrieved with the SurveyMonkey API. The API calls provided the IDs of the choices that people selected. For each survey, we mapped the choice IDs to option A, B, or neither, and then determined whether the abstract chosen was SVD or PubMed, or neither. We tallied the result as well as measuring the consistency of the responses for Survey 56.

### Result

#### Rank analysis

Rankings of the top 10 papers for each of the randomly chosen 9,000 articles made up 90,000 rankings. We looked at how many of these rankings were below 10,000. We evaluated the parameters by looking at the number of articles that were retrieved within the first 10,000 ranks, and determined that TF-IDF matrix with 1000 singular values was the best combination out of all those that we evaluated. The result is shown in **Error! Reference source not found.**. The other 1000 target papers were used for the surveys.

Table 4.1 Number of relevant papers with rankings < 10,000

|  |  |  |  |
| --- | --- | --- | --- |
| NSV | Binary | Term frequency | TF-IDF |
| 200 | 13124 | 15251 | 20360 |
| 400 | 18291 | 19627 | 25943 |
| 600 | 21858 | 22686 | 29752 |
| 800 | 24807 | 24238 | 33090 |
| 1000 | 27106 | 27586 | 35640 |

It is interesting that some top 10 papers recommended by PubMed fell very low, sometimes in millionth, in the ranking by SVD. Table 4.1 lists the five target papers that have the highest average ranks over their top 10 related articles in SVD (TF-IDF, NSV=1000). We examined the titles of these abstracts to understand the cause of the low rankings of supposedly related papers.

Table 4.2 Related papers with low ranks in SVD

|  |  |  |  |
| --- | --- | --- | --- |
| **Target paper** | **Most related paper by PubMed** | **Average rank of top 10** | **Most related paper by SVD** |
| PMID: 12177079  A glossary for health inequalities. | PMID: 17065177  Removing the health domain from the Index of Multiple Deprivation 2004-effect on measured inequalities in census measure of health. | 8268754.5 | PMID: 17308583  Near-field Moiré effect mediated by surface plasmon polariton excitation. |
| PMID: 10738684  An effective algorithm for quick fractal analysis of movement biosignals. | PMID: 18234169  Application of fractal theory in analysis of human electroencephalographic signals | 8268754.5 | PMID: 18003245  Impact analysis of body movement in ambulatory ECG. |
| PMID: 18255845  Parallel implementation of backpropagation neural networks on a heterogeneous array of transputers. | PMID: 7670674  Analysis of training set parallelism for backpropagation neural networks | 8650700.9 | PMID: 18693919  An ontology-based architecture for integration of clinical trials management applications. |
| PMID: 14748427  Education to reduce potassium levels in adolescent, haemodialysis patients. | PMID: 18197864  Fluid compliance among patients having haemodialysis: can an educational programme make a difference? | 9025123.3 | PMID: 18349953  Frequency-modulation spectroscopy with blue diode lasers |

In Table 4.2, the most relevant papers from PubMed are related to the target abstracts, whereas the ones recommended by our SVD-based system seem irrelevant. After examining the abstracts of those papers and the word list we used to annotate them, we found that our word list was missing critical keywords from those targeted abstracts. For example, the words "biosignal", "inequalities", "electroencephalographic", "backpropagation", "haemodialysis", were not present in our word list.

#### Survey analysis

Survey 56, which was sent to everyone as one of the four survey links, received 20 responses. The result is summarized in Table 4.3.

Table 4.3 Result from Survey 56

|  |  |  |
| --- | --- | --- |
| **PubMed** | **SVD** | **Both** |
| 149 | 15 | 55 |

Using the R package irr (Inter-rater reliability) [[66]](https://paperpile.com/c/CvqbpE/oBTZ4), we did an inter-rater reliability analysis for Survey 56 using Fleiss' Kappa [[67]](https://paperpile.com/c/CvqbpE/B1tXM). The Fleiss Kappa is a statistical measure for assessing the agreement among raters when classifying or rating some items, while taking what would be expected by chance into account. There are no standards on significance other than that kappa value is 1 if the raters are in complete agreement. If there is no agreement other than what is expected by chance, the kappa value would be negative. We obtained the kappa value to be 0.267, which shows that there is a fair agreement among the people who did Survey 56.

Out of the 99 surveys that were unique to the 33 respondents, 53 surveys received responses. The aggregated counts across these surveys for PubMed, SVD and the option for both are summarized below in Table 4.4. Out of the total 438 questions that were answered, 294 abstracts were chosen came from PubMed. Therefore, PubMed is significantly better than our SVD-based method (p < 0.00000001).

Table 4.4 Survey result

|  |  |  |
| --- | --- | --- |
| **PubMed** | **SVD** | **Both** |
| 294 - 67.8% | 37 - 8.4 % | 107  - 24.4% |

By looking at some of the survey questions from Survey 56 and Survey 65, we noticed that there were instances where two abstracts were identical or that the difference in their relevancy was small. For some questions, PubMed abstracts were more relevant to the targeted abstracts (**Error! Reference source not found.**) when key terms were missing in the term list and some less relevant terms in the target abstract misguided SVD to retrieve an irrelevant abstract. For others, the abstracts predicted by SVD are relevant to the targeted papers but not as specific as PubMed's suggestions (

Table 4.6). There are also cases where both related papers from PubMed and SVD are judged as equally relevant or irrelevant (Table 4.7). There are a few instances at which the abstracts from SVD were chosen over the ones from PubMed (Table 4.8).

Table 4.5 Abstracts from PubMed were considered more relevant

|  |  |  |  |
| --- | --- | --- | --- |
| **Target paper** | **Most related paper by PubMed** | **Most related paper by SVD** | **keywords** |
| PMID: 8430481 | PMID: 2728297 | PMID: 3704322 | Missing:  Lambs  Affecting: faecal |
| Immunisation of lambs against coccidiosis. | Ovine coccidiosis: heavy infection in young lambs increases resistance without causing disease | Effect of cool storage of faecal samples containing Haemonchus contortus eggs on the results of an in vitro egg development assay to test anthelmintic resistance. |
| PMID: 12577086 | PMID: 9416389 | PMID: 18594170 | Missing:  Affecting:  Complication  penicillin |
| Tetanus | Tetanus: pathophysiology and management | A case of necrotizing fasciitis due to Streptococcus agalactiae, Arcanobacterium haemolyticum, and Finegoldia magna in a dog-bitten patient with diabetes. |
| PMID:19025138 | PMID: 23505784 | PMID: 15798748 |  |
| Scoring the full extent of periodontal disease in the dog: development of a total mouth periodontal score (TMPS) system | Validation of use of subsets of teeth when applying the total mouth periodontal score (TMPS) system in dogs. | Teeth electroexcitability in periodontitis | Dogs |

Table 4.6 Abstracts from SVD were somewhat related to the targeted papers

|  |  |  |  |
| --- | --- | --- | --- |
| **Target paper** | **Most related paper by PubMed** | **Most related paper by SVD** | **keywords** |
| PMID: 21332628  Lipid-altering efficacy and safety profile of combination therapy with ezetimibe/statin vs. statin monotherapy in patients with and without diabetes: an analysis of pooled data from 27 clinical trials | PMID: 15650343  Consistency in efficacy and safety of ezetimibe coadministered with statins for treatment of hypercholesterolemia in women and men | PMID: 20179259  Comparative efficacy and safety of low-dose pitavastatin versus atorvastatin in patients with hypercholesterolemia. | Missing:  coadministered |
| PMID: 9497181  Leptin and body composition of Nigerians, Jamaicans, and US blacks. | PMID: 9098179  Relation between body mass index and body fat in black population samples from Nigeria, Jamaica, and the United States | PMID: 10348494  Collection and interpretation of plasma leptin concentration data in humans. | Missing:  Nigeria, Jamaica, US, United States |
| PMID: 10638908  Neuronal nitric oxide synthase mediates halothane-induced cerebral microvascular dilation. | PMID: 9366472  Region-specific and agent-specific dilation of intracerebral microvessels by volatile anesthetics in rat brain slices. | PMID: 20624383  Relative contribution of eNOS and nNOS to endothelium-dependent vasodilation in the mouse aorta. |  |

Table 4.7 Abstracts from PubMed and SVD were considered equally relevant or irrelevant

|  |  |  |
| --- | --- | --- |
| **Target paper** | **Most related paper by PubMed** | **Most related paper by SVD** |
| PMID: 17947981  Generation of magnetic nonviral gene transfer agents and magnetofection in vitro | PMID: 19301645  Recent advances in magnetofection and its potential to deliver siRNAs in vitro. | PMID: 22826855  Nucleic acid delivery using magnetic nanoparticles: the Magnetofection technology. |
| PMID: 3971960  SCE induction is uncoupled from mutation induction in mammalian cells following exposure to ethylnitrosourea (ENU) | PMID: 6891748  Induction of mutations and sister-chromatid exchanges in Chinese hamster ovary cells by ethylating agents. | PMID: 6889483  Sister chromatid exchange and gene mutation. (ABSTRACT: The parallel induction of sister chromatid exchange (SCE) and single-gene mutation was quantified in Chinese hamster ovary cells …. ) |
| PMID: 10588965  Lack of benefit of a single dose of synthetic human secretin in the treatment of autism and pervasive developmental disorder | PMID: 12553591  Lack of benefit of intravenous synthetic human secretin in the treatment of autism | PMID: 12553591  Lack of benefit of intravenous synthetic human secretin in the treatment of autism |

Table 4.8 Abstracts from SVD were considered more relevant

|  |  |  |
| --- | --- | --- |
| **Target paper** | **Most related paper by PubMed** | **Most related paper by SVD** |
| PMID: 19307985  Early lipopolysaccharide-induced reactive oxygen species production evokes necrotic cell death in human umbilical vein endothelial cells | PMID: 15569826  Atrial natriuretic peptide induces mitogen-activated protein kinase phosphatase-1 in human endothelial cells via Rac1 and NAD(P)H oxidase/Nox2-activation | PMID: 21565835  Transient receptor potential melastatin 4 inhibition prevents lipopolysaccharide-induced endothelial cell death. |
| PMID: 23399035  The modulation of hepatitis C virus 1a replication by PKR is dependent on NF-kB mediated interferon beta response in Huh7.5.1 cells | PMID: 19840259  Double-stranded RNA-activated protein kinase inhibits hepatitis C virus replication but may be not essential in interferon treatment. | PMID: 23107224  The kinase inhibitor Sorafenib impairs the antiviral effect of interferon α on hepatitis C virus replication. |
| PMID: 12410954  A case-control study on risk factors of helicobacter pylori infection in out-patients with stomach diseases. | PMID: 11960066  The incidence of Helicobacter pylori infection is not increased among obese young individuals in Greece. | PMID: 9143215  Association of Sauropus androgynus and bronchiolitis obliterans syndrome: a hospital-based case-control study. |

### Discussion

By looking at both the distributions of ranks of the related papers from PubMed followed by evaluation from people, we examined how well SVD retrieve relevant abstracts from 13-million abstract corpus. We used the rank analysis to determine which weighting scheme and NSV would be the best parameters for predictions. Just like in Chapter 2, we used PubMed as the gold standard to help us determine the parameters for SVD in this experiment. In the surveys, we then compared the related articles from SVD with that of PubMed.

In the analysis with survey evaluation, we gave people unique surveys so that we could evaluate many abstracts and avoid the need to resolve different answers to the same questions. We added the third option ("Both are relevant OR Both are irrelevant") to account for the scenarios when people could not choose either A or B with full confidence. During an earlier sanity check on our rank analysis, we knew that there were a few instances when the top-ranked related paper from SVD was identical to that of PubMed, as an example in Table 4.7. Without the third option, we would need a large number of responses to for those target papers and determine that the two abstracts are equally relevant or irrelevant when the responses are equally split.

To ensure that quality of the survey responses was based on good judgment, we let the respondents skip over the questions if they did not fully understand the abstracts. Moreover, Survey 56, which everyone had a chance to answer, gave us a sense to the consistency of people's opinions. The kappa value was not to the degree of complete agreement. However, its positive value indicated a level of agreement above chance.

When an abstract from SVD was chosen over one from PubMed, it did not always mean that PubMed's abstract was irrelevant to the target abstract. Abstracts from PubMed and SVD in Table 4.7 and Table 4.8 showed that both methods retrieved relevant articles. When an abstract from SVD was preferable over an abstract from PubMed, it was often because its title text contained more relevant keywords. When both SVD and PubMed abstracts were considered equally relevant, their titles often did not have all the keywords from the target abstract's title while their abstract texts highly related to the target abstract. The choice between "Both" or "SVD" might depend on whether the respondents read all the text or based their decision mainly on the title text.

Sometimes SVD did not quite retrieve an article on the exact topic, possibly due to missing words. In the second examples of

Table 4.6, SVD could not retrieve an abstract that matches the target abstract closely because our word list did not contain geographical or demographic words. The third example in **Error! Reference source not found.** shows that SVD was able to retrieve an article on periodontitis but not about the disease in dogs, possibly because the word list did not contain “dogs”.

The survey result gave us an idea of how well SVD retrieve relevant articles comparing to PubMed. Creating the surveys also allowed us to examine the abstracts from SVD. From several abstracts, it was evident that missing words in the original word list affect the quality of the retrieved abstracts. Non-biomedical words, such as those on geographic location, demographics, were often pertinent to the topic of a biomedical abstract. To improve the performance of SVD, we would need to include more terms in the word list that creates our document-by-term matrix. Moreover, it might be helpful to put more emphasis on the title text.

The survey result showed that the abstracts from PubMed were preferable two third of the time. However, 24% of the answered questions had abstracts that were considered equivalently relevant to the targeted abstracts, and 8% of the time the abstracts from SVD were considered more relevant than that of PubMed. Despite SVD being a completely automated process that used only abstract and title text of the papers, its predictions were indistinguishable from that of PubMed in one-third of the questions. While PubMed is the clear winner, its predictions require manual MeSH terms assignments by annotators who have access to the entire text of the articles. Perhaps with further improvement on the method with SVD, we can achieve a better trade-off between quality of retrieval and the number man-hours that go into making the predictions.

## Discussion

With the increasing number of publications, continuing advances in the text retrieval technology is important to help researchers discover relevant information for their work. Since PubMed, the most popular search engine for biomedical literature, relies on manual annotations to pre-compute their related articles result, we were motivated to test how well an unsupervised machine learning method can retrieve similar biomedical articles. In the study, we were able to demonstrate the text retrieval performance using LSA, an SVD-based technique, with different parameters. We tested the parameters on smaller abstract corpora as well as a much larger corpus. We showed that sampling the matrix can help make an educated estimate on the number of singular values in a reduced SVD. Finally, we validated the performance with evaluation from students with knowledge of the biomedical science and experience of reading biomedical literature.

While the result from the surveys showed that PubMed may provide more relevant articles than SVD can, we also found that SVD could retrieve related abstracts that were considered equally good or better than PubMed’s suggestions in one-third of the survey questions. Such performance is notable, since our SVD-based method only took in the title and abstract text as input. The remaining predictions were made based on automated computation. PubMed’s related paper recommendations, on the other hands, rely on indexers to read papers and label them with MeSH controlled vocabulary. These specialists who analyze the content and index the concepts of each article hold at least a bachelor in biomedical science. In other words, not only that articles need to be labelled before it becomes available in the search result, but the process also requires expensive manual work. Finding a more automated way to retrieve relevant articles would save not only time but also the tremendous human resources from highly trained scientists and make new publications available in less time. It could also be useful for organizations that do not have the same human resources as PubMed. Our method is one step toward that goal, and there are a few ways to enhance the method further.

For example, we suspect that our method can be improved by including more relevant terms that can be important to the topics of biomedical articles. We tested different distance metric and term weighting scheme, but have not examined the effect of the word list to our predictions. We saw instances in the survey questions where SVD's recommendation was subpar to that of PubMed when some keywords of the target abstracts were missing from the terms list. The list came from five major groups out of the 15 semantic groups in the UMLS: Anatomy, Disorder, Genes & Molecular Sequences, Physiology, and Chemical & Drugs. There are other semantic groups, such as Geographic Areas and Occupations, that might have improved the quality of relevant abstracts that were retrieved if they were part of our terms list. It is also likely that there are words that are not part of UMLS but are critical concepts that define some of the articles.

Our document-by-term matrices were constructed from the title and abstract of the papers, whereas PubMed's recommendation is based manually annotated keywords that summarize the concepts of the entire articles. Including the full text might also affect the text retrieval performance whenever full text is available. Since most people use title text to judge relevancy, weighting title text differently from abstract and body text can also be helpful.

Others have looked into using known associations of words to boost the topics in document-term matrix. Document padding or sprinkling, as described in previous studies [[68,69]](https://paperpile.com/c/CvqbpE/5VFv6+xJAn0), could provide a way of performing partially supervised learning in SVD. One of them used this technique to help scoring association between biomedical concepts [[69]](https://paperpile.com/c/CvqbpE/xJAn0), and the other one used the technique on document classification. Both studies used existing knowledge from biomedical databases, UMLS Metathesaurus and Biomedical Pathways Databases, to augment the document matrix by adding the artificial groups of terms to as terms that occur in the same documents. They suggested that this strengthens the associations among the words of the same topics. It would be interesting to know whether document padding can help with retrieving related biomedical articles.

Besides improving the results, there are other areas such as computation overheads and user experience that are also important parts of a recommendation system.

In our study, the 13 million abstract corpus takes about half a week to run through SVD, given sufficient memory and 40 cores. The text retrieval based on the decomposed matrices can then only retrieve papers that were in the original matrix. SVD is computationally expensive and thus redoing it whenever new papers are added to the matrix is intractable. Given that papers are constantly being published, there needs to be a faster way to add new papers into the system. One area that can be explored is using incremental SVD to make sure the system is scalable and can be efficiently updated. Some have studied methods of incrementally building an SVD model and revising it when data is added or removed [[28,70]](https://paperpile.com/c/CvqbpE/TIPDS+D9isl). This can be done by applying a fold-in technique, which uses the existing decomposed matrix of the row dimension to approximate the new information for the decomposed matrix of the column dimension, or vice versa. After some number of folding-in, SVD would need to be redone on the new matrix to ensure the loss of quality does not affect the recommendations.

To make a recommendation system popular and useful to people, algorithms are only part of the solution. In his keynote address at the 2009 ACM Conference on Recommender Systems, Martin [[71]](https://paperpile.com/c/CvqbpE/mFV0m)] explained that there are work to be done on addressing the problems that make up the whole user experience besides refining the algorithms that we already have. These include presenting the data in a sensible and manageable way and explaining the recommendations. The latter is especially important for our SVD-based method, which is an unsupervised machine learning technique that has been applied in knowledge discovery to uncover previously unknown associations between concepts. It is likely that our SVD can recommend articles that might not seem relevant to our target article, but is related somehow through other concepts. Therefore, it would be useful for users to understand why SVD consider an article related to another article.

## Bibliography

1. [Rebholz-Schuhmann D, Oellrich A, Hoehndorf R. Text-mining solutions for biomedical research: enabling integrative biology. Nat Rev Genet. 2012;13: 829–839.](http://paperpile.com/b/CvqbpE/fGjoS)

2. [Hristovski D, Rindflesch T, Peterlin B. Using literature-based discovery to identify novel therapeutic approaches. Cardiovasc Hematol Agents Med Chem. 2013;11: 14–24.](http://paperpile.com/b/CvqbpE/ScRJO)

3. [Swanson DR. Fish oil, Raynaud’s syndrome, and undiscovered public knowledge. Perspect Biol Med. 1986;30: 7–18.](http://paperpile.com/b/CvqbpE/0mjse)

4. [Rodriguez-Esteban R. Biomedical text mining and its applications. PLoS Comput Biol. 2009;5: e1000597.](http://paperpile.com/b/CvqbpE/EgtGW)

5. [Rejto P. Knowledge discovery and data mining in pharmaceutical cancer research. Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM; 2011. pp. 781–781.](http://paperpile.com/b/CvqbpE/vXRTL)

6. [Clark S. Vector Space Models of Lexical Meaning. The Handbook of Contemporary Semantic Theory. John Wiley & Sons, Ltd; 2015. pp. 493–522.](http://paperpile.com/b/CvqbpE/Y49Aw)

7. [Christopher D. Manning, Prabhakar Raghavan, Hinrich Schütze. An Introduction to Information Retrieval. Cambridge University Press; 2009.](http://paperpile.com/b/CvqbpE/oXPGi)

8. [Uddin MM, Hassan MT, Karim A. Personalized versus non-personalized tag recommendation: A suitability study on three social networks. 2011 IEEE 14th International Multitopic Conference. 2011. pp. 56–61.](http://paperpile.com/b/CvqbpE/NuSKq)

9. [Poriya A, Patel N. Non-Personalized Recommender Systems and User- based Collaborative Recommender Systems. International Journal of Applied Information Systems (IJAIS). 2014;](http://paperpile.com/b/CvqbpE/4fZqb)

10. [Rich E. User modeling via stereotypes. Cogn Sci. Wiley Online Library; 1979;3: 329–354.](http://paperpile.com/b/CvqbpE/N4Fxn)

11. [Lops P, de Gemmis M, Semeraro G. Content-based Recommender Systems: State of the Art and Trends. In: Ricci F, Rokach L, Shapira B, Kantor PB, editors. Recommender Systems Handbook. Springer US; 2011. pp. 73–105.](http://paperpile.com/b/CvqbpE/Jgiep)

12. [Isinkaye FO, Folajimi YO, Ojokoh BA. Recommendation systems: Principles, methods and evaluation. Egyptian Informatics Journal. 2015;16: 261–273.](http://paperpile.com/b/CvqbpE/yYbuI)

13. [Ekstrand MD, Riedl JT, Konstan JA. Collaborative Filtering Recommender Systems. Foundations and Trends® in Human–Computer Interaction. Now Publishers; 2011;4: 81–173.](http://paperpile.com/b/CvqbpE/KsnSi)

14. [Goldberg D, Nichols D, Oki BM, Terry D. Using Collaborative Filtering to Weave an Information Tapestry. Commun ACM. New York, NY, USA: ACM; 1992;35: 61–70.](http://paperpile.com/b/CvqbpE/gBJoD)

15. [Resnick P, Iacovou N, Suchak M, Bergstrom P, Riedl J. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work. New York, NY, USA: ACM; 1994. pp. 175–186.](http://paperpile.com/b/CvqbpE/BXgwG)

16. [Ed.D FL, Joy D, Schaefer H-P, Charron A. OWL: A Recommender System for Organization-Wide Learning. Journal of Educational Technology & Society. International Forum of Educational Technology & Society; 2000;3: 62–76.](http://paperpile.com/b/CvqbpE/htgbH)

17. [Bell RM, Koren Y. Lessons from the Netflix Prize Challenge. SIGKDD Explor Newsl. New York, NY, USA: ACM; 2007;9: 75–79.](http://paperpile.com/b/CvqbpE/Vnn0S)

18. [Bennett J, Lanning S. Netflix: The Netflix Prize. KDD Cup and Workshop in conjunction with KDD. 2007.](http://paperpile.com/b/CvqbpE/J2ZAZ)

19. [Koren Y. The bellkor solution to the netflix grand prize. Netflix prize documentation. 2009;81: 1–10.](http://paperpile.com/b/CvqbpE/Cug8r)

20. [Gower S. Netflix Prize and SVD. 2014; Available:](http://paperpile.com/b/CvqbpE/CpiVm) <http://buzzard.ups.edu/courses/2014spring/420projects/math420-UPS-spring-2014-gower-netflix-SVD.pdf>

21. [Koren Y, Bell R, Volinsky C. Matrix factorization techniques for recommender system. IEEE. 2009;](http://paperpile.com/b/CvqbpE/MeCEk)

22. [Sarwar BM, Karypis G, Konstan JA, Riedl JT. Application of Dimensionality Reduction in Recommender System -- A Case Study. ACM WebKDD 2000 Workshop. 2000.](http://paperpile.com/b/CvqbpE/Mbf5W)

23. [Baker K. Singular Value Decomposition Tutorial. 2005; Available:](http://paperpile.com/b/CvqbpE/JCpRd) <http://home.iitk.ac.in/~crkrish/MLT/PreRequisites/linalgWithSVD.pdf>

24. [Wall ME, Rechtsteiner A, Rocha LM. Singular Value Decomposition and Principal Component Analysis. In: Berrar DP, Dubitzky W, Granzow M, editors. A Practical Approach to Microarray Data Analysis. Springer US; 2003. pp. 91–109.](http://paperpile.com/b/CvqbpE/R0Gt3)

25. [Berry MW, Dumais ST, O’Brien GW. Using Linear Algebra for Intelligent Information Retrieval. SIAM Rev. 1995;37: 573–595.](http://paperpile.com/b/CvqbpE/7QD0q)

26. [Holmes M, Gray A, Isbell C. Fast SVD for large-scale matrices. Workshop on Efficient Machine Learning at NIPS. 2007. pp. 249–252.](http://paperpile.com/b/CvqbpE/HffSf)

27. [Baker CG, Gallivan KA, Van Dooren P. Low-rank incremental methods for computing dominant singular subspaces. Linear Algebra Appl. 2012;436: 2866–2888.](http://paperpile.com/b/CvqbpE/VoEcy)

28. [Sarwar B, Karypis G, Konstan J. Incremental Singular Value Decomposition Algorithms for Highly Scalable Recommender Systems. Fifth International Conference on Computer and Information Science. 2002;](http://paperpile.com/b/CvqbpE/TIPDS)

29. [PMC Open Access Subset [Internet]. [cited 11 Dec 2016]. Available:](http://paperpile.com/b/CvqbpE/SXmod) <https://www.ncbi.nlm.nih.gov/pmc/tools/openftlist/>

30. [Salakhutdinov R, Mnih A. Probabilistic matrix factorization. NIPS. 2011. pp. 1–8.](http://paperpile.com/b/CvqbpE/wMryQ)

31. [Chen H, Martin B, Daimon CM, Maudsley S. Effective use of latent semantic indexing and computational linguistics in biological and biomedical applications. Front Physiol. 2013;4: 8.](http://paperpile.com/b/CvqbpE/0mZO7)

32. [Zhu S, Zeng J, Mamitsuka H. Enhancing MEDLINE document clustering by incorporating MeSH semantic similarity. Bioinformatics. 2009;25: 1944–1951.](http://paperpile.com/b/CvqbpE/uj0fY)

33. [Liu H, Rastegar-Mojarad M. Literature-Based Knowledge Discovery. Big Data Analysis for Bioinformatics and Biomedical Discoveries. CRC Press; 2016; 233.](http://paperpile.com/b/CvqbpE/qmGxs)

34. [Shatkay H, Feldman R. Mining the biomedical literature in the genomic era: an overview. J Comput Biol. 2003;10: 821–855.](http://paperpile.com/b/CvqbpE/JZTOL)

35. [Bodenreider O. The Unified Medical Language System (UMLS): integrating biomedical terminology. Nucleic Acids Res. 2004;32: D267–70.](http://paperpile.com/b/CvqbpE/ds0q4)

36. [Download MEDLINE/PubMed Data [Internet]. U.S. National Library of Medicine; 6 Dec 2016 [cited 11 Dec 2016]. Available:](http://paperpile.com/b/CvqbpE/WMOwR) <https://www.nlm.nih.gov/databases/download/pubmed_medline.html>

37. [John Wilbur W. Modeling Text Retrieval in Biomedicine. In: Chen H, Fuller SS, Friedman C, Hersh W, editors. Medical Informatics. Springer US; 2005. pp. 277–297.](http://paperpile.com/b/CvqbpE/yRxMv)

38. [Maron ME, Kuhns JL. On Relevance, Probabilistic Indexing and Information Retrieval. J ACM. New York, NY, USA: ACM; 1960;7: 216–244.](http://paperpile.com/b/CvqbpE/DFBH1)

39. [Rocchio J. Relevance feedback information retrieval, G. Salton ed., The Smart retrieval system| experiments in automatic document processing, p 313-323. Prentice-Hall, Englewood Cli s, NJ; 1971.](http://paperpile.com/b/CvqbpE/gq5ih)

40. [Sparck Jones K. A Statistical Interpretation of Term Specificity and Its Application In Retrieval. Journal of Documentation. 1972;28: 11–21.](http://paperpile.com/b/CvqbpE/GlfCH)

41. [Lin J, Wilbur WJ. PubMed related articles: a probabilistic topic-based model for content similarity. BMC Bioinformatics. 2007;8: 423.](http://paperpile.com/b/CvqbpE/Aw6nU)

42. [Harter SP. A probabilistic approach to automatic keyword indexing. Part II. An algorithm for probabilistic indexing. J Am Soc Inf Sci. Wiley Subscription Services, Inc., A Wiley Company; 1975;26: 280–289.](http://paperpile.com/b/CvqbpE/4ayxx)

43. [Hersh W, Cohen A, Yang J, Bhupatiraju RT, Roberts P, Hearst M. TREC 2005 Genomics Track Overview.](http://paperpile.com/b/CvqbpE/TwrOc)

44. [Indexing with MeSH Vocabulary [Internet]. U.S. National Library of Medicine; 1 Mar 2001 [cited 29 Nov 2016]. Available:](http://paperpile.com/b/CvqbpE/BWJBb) <https://www.nlm.nih.gov/bsd/disted/pubmedtutorial/015_030.html>

45. [Gipp B, Beel J, Hentschel C. Scienstein: A research paper recommender system. Proceedings of the international conference on Emerging trends in computing (ICETiC’09). 2009. pp. 309–315.](http://paperpile.com/b/CvqbpE/fnCVk)

46. [Giles CL, Bollacker KD, Lawrence S. CiteSeer: An Automatic Citation Indexing System. Proceedings of the Third ACM Conference on Digital Libraries. New York, NY, USA: ACM; 1998. pp. 89–98.](http://paperpile.com/b/CvqbpE/0WnKY)

47. [Lee J, Lee K, Kim JG. Personalized Academic Research Paper Recommendation System. SRS’15. 2015; Available:](http://paperpile.com/b/CvqbpE/oW61J) <http://www.comp.hkbu.edu.hk/~fwang/files/srs_2015_papers/srs2015_paper_1.pdf>

48. [Sugiyama K, Kan M-Y. Scholarly Paper Recommendation via User’s Recent Research Interests. Proceedings of the 10th Annual Joint Conference on Digital Libraries. New York, NY, USA: ACM; 2010. pp. 29–38.](http://paperpile.com/b/CvqbpE/4yor7)

49. [Read by QxMD. In: QxMD Medical Apps [Internet]. [cited 7 Dec 2016]. Available:](http://paperpile.com/b/CvqbpE/DrMLt) <http://www.qxmd.com/apps/read-by-qxmd-app>

50. [Master Thesis UBC [Internet]. GitHub; [cited 11 Dec 2016]. Available:](http://paperpile.com/b/CvqbpE/CHnUe) <https://github.com/santina/Master_Thesis_UBC>

51. [MEDLINE, PubMed, and PMC (PubMed Central): How are they different? [Internet]. U.S. National Library of Medicine; 6 Jan 2016 [cited 29 Nov 2016]. Available:](http://paperpile.com/b/CvqbpE/7Lr26) <https://www.nlm.nih.gov/pubs/factsheets/dif_med_pub.html>

52. [Joseph Gonzalez YL. GraphLab collaborative filtering library: efficient probabilistic matrix/tensor factorization on multicore [Internet]. [cited 11 Dec 2016]. Available:](http://paperpile.com/b/CvqbpE/TFeWE) <http://select.cs.cmu.edu/code/graphlab/pmf.html>

53. [Sayers E. E-utilities Quick Start. National Center for Biotechnology Information (US); 2013.](http://paperpile.com/b/CvqbpE/say7b)

54. [Bird S, Klein E, Loper E. Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit. “O’Reilly Media, Inc.”; 2009.](http://paperpile.com/b/CvqbpE/SJBqP)

55. [Tsuruoka Y, Tateishi Y, Kim J-D, Ohta T, McNaught J, Ananiadou S, et al. Developing a Robust Part-of-Speech Tagger for Biomedical Text. In: Bozanis P, Houstis EN, editors. Advances in Informatics. Springer Berlin Heidelberg; 2005. pp. 382–392.](http://paperpile.com/b/CvqbpE/bZfQa)

56. [Gonzalez JE, Low Y, Gu H, Bickson D. Powergraph: Distributed graph-parallel computation on natural graphs. Presented as part of the. usenix.org; 2012; Available:](http://paperpile.com/b/CvqbpE/FwWz7) <https://www.usenix.org/conference/osdi12/technical-sessions/presentation/gonzalez>

57. [Hern V, Rom AJE, Tom an A. Restarted Lanczos Bidiagonalization for the SVD in SLEPc. Available:](http://paperpile.com/b/CvqbpE/K2hQO) <http://slepc.upv.es/documentation/reports/str8.pdf>

58. [Weisstein EW. Frobenius norm. In: MathWorld--A Wolfram Web Resource [Internet]. Wolfram Research, Inc.; 2017. Available:](http://paperpile.com/b/CvqbpE/om6qV) <http://mathworld.wolfram.com/FrobeniusNorm.html>

59. [Mattingley J. Low-rank approximation of a matrix [Internet]. 23 Sep 2011 [cited 13 Jan 2017]. Available:](http://paperpile.com/b/CvqbpE/YTvgs) <https://inst.eecs.berkeley.edu/~ee127a/book/login/l_svd_low_rank.html>

60. [Jones E, Oliphant T, Peterson P, Others. SciPy: Open source scientific tools for Python [Internet]. Available:](http://paperpile.com/b/CvqbpE/D6fqV) <http://www.scipy.org/>

61. [Fung WS, Hariharan R, Harvey NJA, Panigrahi D. A general framework for graph sparsification. Proceedings of the forty-third annual ACM symposium on Theory of computing. ACM; 2011. pp. 71–80.](http://paperpile.com/b/CvqbpE/saXVC)

62. [Gakkhar S. sitag: TrackRank. In: GitHub [Internet]. [cited 23 Jan 2017]. Available:](http://paperpile.com/b/CvqbpE/16SNn) <https://github.com/sitag/TrackRank>

63. [SurveyMonkey [Internet]. [cited 24 Jan 2017]. Available:](http://paperpile.com/b/CvqbpE/2XQo1) <https://www.surveymonkey.com/>

64. [API Docs | SurveyMonkey API Developer Portal [Internet]. [cited 21 Jan 2017]. Available:](http://paperpile.com/b/CvqbpE/CcPHS) <https://developer.surveymonkey.com/api/v3/>

65. [Hester J. gmailr: Access the Gmail RESTful API [Internet]. 2016. Available:](http://paperpile.com/b/CvqbpE/IxKDu) [https://CRAN.R-project.org/package=gmailr](https://cran.r-project.org/package=gmailr)

66. [Gamer M, Lemon J, <puspendra.pusp22@gmail.com> IFPS. irr: Various Coefficients of Interrater Reliability and Agreement [Internet]. 2012. Available:](http://paperpile.com/b/CvqbpE/oBTZ4) [https://CRAN.R-project.org/package=irr](https://cran.r-project.org/package=irr)

67. [Fleiss JL, Cohen J. The equivalence of weighted kappa and the intraclass correlation coefficient as measures of reliability. Educ Psychol Meas. Sage Publications Sage CA: Thousand Oaks, CA; 1973;33: 613–619.](http://paperpile.com/b/CvqbpE/B1tXM)

68. [Yang H, King I. Sprinkled Latent Semantic Indexing for Text Classification with Background Knowledge. Lecture Notes in Computer Science. 2009. pp. 53–60.](http://paperpile.com/b/CvqbpE/5VFv6)

69. [Abate F, Ficarra E, Acquaviva A, Macii E. Improving Latent Semantic Analysis of Biomedical Literature Integrating UMLS Metathesaurus and Biomedical Pathways Databases. Biomedical Engineering Systems and Technologies. Springer, Berlin, Heidelberg; 2011. pp. 173–187.](http://paperpile.com/b/CvqbpE/xJAn0)

70. [Brand M. Fast online SVD revisions for lightweight recommender systems. Proceedings of the 2003 SIAM International Conference on Data Mining. pp. 37–46.](http://paperpile.com/b/CvqbpE/D9isl)

71. [Martin FJ. Recsys’09 industrial keynote: top 10 lessons learned developing deploying and operating real-world recommender systems. Proceedings of the third ACM conference on Recommender systems. ACM; 2009. pp. 1–2.](http://paperpile.com/b/CvqbpE/mFV0m)