Relatedness Ranking of Biomedical Publications Using Figure Content

Luke Hansen (B.S. Computer Science & Mathematics ’15),
with Mate Nagy (PhD Candidate, CB&B),
and Professor Michael Krauthammer (Yale Med. School)
April 29, 2015

Abstract

As the domain of biomedical publications continues to grow, sophisticated information retrieval engines will play an increasingly important role for scientists and medical practitioners to access the literature needed for their work. Because the experiments and results of biomedical research are often presented as images, and because this information is often not fully and/or explicitly declared in the text, advanced biomedical image search engines should prove powerful tools for retrieving experiments, medical results, protocols, and other biomedical information encoded in image form. The Yale Image Finder is one of such biomedical image search engines and allows users to search based on: text embedded in an image itself (parsed using optical character recognition); text in the image’s caption; and text in the associated article’s title or abstract. Unique to our tool is also a related images feature, which allows researchers and doctors alike to retrieve similar experiments and results (and their associated articles), given an image of interest. This feature has been the focus of my CS 490 project this year. Our method for providing such related images involves first determining the subset of articles related to that of the given image; and subsequently determining related images from this subset. Using only textual analysis so far, we can rapidly and accurately establish closely related biomedical images. In addition to various case studies, we explore several metrics for evaluating the advantages of our relatedness feature over traditional tools. Our results indicate that the inclusion of figure information significantly alters relatedness rankings and may provide more specific results. Immediate future work will include further evaluation of our results based on actual user sessions, as well as construction of a comprehensive feature discrimination system using Haralick feature analysis and a convolutional neural network for content-based image retrieval.
1 Note

The research described here was continued from Fall 2014. This report focuses primarily on the work completed Spring 2015, with only the necessary background for cogency. Methods, results, and conclusions from Fall 2014 can be found in the attached Supplementary Materials.

2 Introduction

The Yale Image Finder (YIF) is a publicly accessible tool developed in the lab of Professor Michael Krauthammer (Yale School of Medicine) to retrieve open-access biomedical images and their associated articles. The web app initiates searches based on text embedded in the image itself, as well as in the image’s caption, the paper’s title, and the paper’s abstract. In addition, a beta version of our tool provides a feature for retrieving related images (along with their associated articles) given an image of interest (which could come from a search result or simply from browsing any paper in our corpus). A version of the related images feature has also recently been implemented in an A/B testing environment by Elsevier, a major biomedical publisher.

From a survey of 31 scientists and physicians (data not shown), we concluded that such a related images feature could be quite useful for: retrieving related experiments and techniques; validating or comparing results; finding higher caliber literature (e.g., from a more familiar author or institution); and overall literature reviews. In addition, 90% of those surveyed indicated that the end goal in accessing related images would often be to retrieve the underlying articles those images are found in (as opposed to simply viewing the image without the remainder of the article). With these user comments in mind, we set about to build an optimal tool for retrieving related biomedical images.

The initial implementation of the related images feature (built before I joined the lab) retrieved similar images based solely on text found in the image’s caption. In our most recent version of the tool, we first calculate a subset of articles related to that of the image in question, based on the paper’s authors, title, abstract, and journal; from this subset of papers, we then determine the top related images (and their underlying articles) utilizing image captions and embedded text (see Methods). The motivation for this two-fold approach came largely from our survey, which indicated that users were most interested in the articles underlying a related image; hence conceptually, it made sense to reduce our search for related images to a subset of related articles.

In order to evaluate the merits of our novel related images feature, we devised numerous methods to determine the effect of including figure features in calculating relatedness, as well as to indicate whether this effect actually improved the feature. These metrics all attempt to compare relatedness as calculated solely by article features to relatedness as calculated by article and figure features together. Because our tool incorporates both article and figure attributes, while conventional relatedness features incorporate only article fields, such metrics should form a basis upon which we can analyze the qualities of our method. We specifically explore correlation and clustering analyses. The preliminary results indicate that our tool produces significantly varied rankings as compared to conventional methods, and that these
rankings may be superior in terms of relatedness.

While our current version of the YIF and its related images feature should prove useful, the ideal image finder (and its related images feature) would have the capability to search by features within an image itself, termed content-based image retrieval (CBIR). Our first stage of feature analysis is nearly complete (implemented by Mate Nagy), and is based on Haralick textural features [5]. In this paper, Haralick et. al. detail 14 statistics that attempt to describe the texture and tone of an image. This process should prove especially useful for our related images feature: in querying for related images, we can include similar Haralick features as a new basis for relatedness. Finally, another goal of this project is to implement a convolutional neural network (CNN) in an attempt to learn the finest details of relatedness.

3 Methods

3.1 Apache Lucene

We power our search engine with Apache Lucene, a “high-performance, full-featured text search engine library written entirely in Java,” [1]. In Lucene, a searchable index is built up by a number of “documents”, where a document contains various fields (such as author, title, abstract, etc.). In our case, each Lucene “document” is actually a figure – this allows us to link figure-specific information such as its caption or any embedded text (recognized using optical character recognition [OCR] implemented by Mate Nagy), to the image itself; unfortunately it also involves the duplication of general article attributes such as title, abstract, etc. (For future iterations of the tool, we consider rebuilding the index such that our Lucene documents represent an article as opposed to a figure; we then could include each figure as a field of the article.) A Lucene query works by specifying various search terms, the document field(s) to search upon, and a ‘boost’ factor to weight one query over another when searching with multiple queries. The results of a search query are returned in order of their ‘score’, which is determined by how closely the document matches the original queries (each weighted by their boost factor). A Lucene score is based on cosine similarity; further details can be found here: http://lucene.apache.org/core/3_6_2/scoring.html.

3.2 Relatedness Calculation

In general, the related figures feature works by retrieving data for a given image, and constructing Lucene queries for other figures based on this data. Lucene provides a convenient class, MoreLikeThis, which has methods to construct a query out of the top tf*idf terms in a given field of a given document (again, in our case, a “document” is really a figure with the figure’s info in addition to the underlying article’s info). A tf*idf (term frequency * inverse document frequency) score is a statistic commonly used to measure the importance and uniqueness of a term to a particular document within a corpus. High tf*idf scores indicate a term is simultaneously prevalent in a particular article, but sparse in the entire corpus. We want to use high tf*idf terms when searching for related figures, since presumably these will retrieve figures in the same niche topic as the source figure. Thus we can create a similarity query for a particular figure in the following manner: first create a subquery for each field to
search across (e.g., authors, title, abstract); next use the MoreLikeThis class to retrieve the
top tf*idf terms from each of the given figure’s fields; finally boost each term in the query
by its tf*idf score, such that other figures with the same high tf*idf term in the same field
will receive high relatedness scores. The last thing to note is that an entire field’s query can
also be boosted; for example, since a figure’s caption likely provides more information about
the figure than simply the title, we can weight the overall caption query more heavily than
the title query.

In the first stage of building out the related images tool, we simply included various
document fields in our search query, and subjectively determined the overall boosts to give
each query. This was a very iterative process, though there were a few interesting aspects
that we implemented: for example, the authors listed on a paper generally contributed to
the paper in the order that they are listed, with the exception that the last author is usually
reserved for the Principal Investigator (PI) of the lab. The authors of a paper are certainly a
decent metric for finding related images; we further hypothesized that the contributions by
author should also be taken into account when querying. Using Lucene’s convenient boost
features, we were able to decrease the weight of each author in the order that they appear,
while maintaining a higher boost for the PI, as she may serve as a proxy for the topic of a
paper. Another small aspect we implemented was limiting the results to figures from articles
that appeared within 10 years of the source article.

Perhaps the feature that resulted in the largest improvement in performance, however,
was separating the search process into two stages (as described briefly in the Introduction).
With this scheme, we isolate the underlying article fields (e.g., authors, title, or abstract);
we search for articles (not figures) based on these features, providing a context radius; and
then we subsequently search for figures within the context radius using the figure-specific
fields (currently only the image caption and any OCR-detected text). The similarity score
(and thus the ranking) that we assign to related figures (and their underlying articles) is
calculated as the score of the related article multiplied by the score of the related figure from
that article.

3.3 Caching Results

Because of the inherently applied nature of our tool, the key parameter to optimize is us-
ability. (This aspect is reflected in the subjective nature of relatedness to begin with.) We
noticed queries for related articles took a relatively long time to run (5 - 10 seconds), and
hence decided to cache our related results for every image in our corpus. The advantage
of caching is a tremendous speed increase, however the caching process itself took 4 days
to process and would miss any articles added to the corpus after a caching (hence caching
should be updated once a month or so).

To implement our caching procedure, we parallelized the process across 16 cores. This is
implemented in ImgManager.java and can be run with the shell script runProcessArticles.sh.

3.4 Evaluation Metrics

As mentioned in the Introduction, we wanted to determine if the underlying articles found
through our related images feature significantly differed from related articles found through
conventional methods, namely those using only article features; and moreover we wanted to
determine if the results from our tool represented an improved ranking of related articles.
In the following metrics, we use the first stage of our two-part method as an example of a
relatedness calculation incorporating only article information, and we use the full two-part
method as an example of a relatedness calculation incorporating both article and figure
information. We refer to the former (article features only) as the “article query”, while the
latter (article and figure features together) is the “figure query.”

We first looked at the correlation of the ranking vectors returned by both methods. For
a given figure, the article query will return an ordering of the top \( n \) related articles. The
figure query then returns its own ordering from these \( n \) articles; thus in essence, the figure
query re-ranks the article query. Because the rankings contain the same set of articles, we
can simply correlate the ranking vectors. We did this for 100 randomly selected figures from
our corpus to see overall how significantly the figure query re-ranks the article query (see
ArticleFigureOrder.java and articleFigureOrder.R). From these 100 figures, we then exam-
ined the extreme cases: we looked at figures that had minimal correlation (i.e., significant
re-ranking), and those that had the highest correlation (i.e., very little re-ranking). In ad-
dition to the correlation between the two vectors, we can also simply calculate the overlap
between the top 20 results. We may even prefer this simpler score, since we hypothesize that
a user will generally not look through more than 20 results (even that is fairly high).

Next we applied this correlation technique to all the figures from a given article. Looking
at a few case studies, we plotted the rankings of the figure query against those of the article
query (and looked at the correlation and top-20 overlap as well). This would identify any
figures within an article that caused an extreme re-ranking (or alternatively very little re-
ranking), and potentially shed some light as to what causes such re-ranking.

Lastly, we implemented cluster analysis to compare how the article and figure queries
group related articles. Given a set of articles, we create a similarity matrix based on the
article query or figure query as follows:

**Article query:** (implemented across ArticleGraph.java and ClusterArticle.R)

For each article in the set:

- Calculate the related articles (and their corresponding Lucene score) by the article
  query (many of the least related articles will likely have a score of 0)

Define a similarity matrix \( X \) where \( X[i,j] = \text{mean of article } j \text{'s score as ranked by article } i \text{ and article } i \text{'s score as ranked by article } j \) (either of which could be 0)

(Note that the score associated with article \( i \) as ranked by \( j \) and the score associated
with article \( j \) as ranked by \( i \) are not necessarily equal.)

To create such a similarity matrix by the figure query, we first must note that the figure
query returns related figures, or their underlying related articles, for a given figure of inter-
est. Hence to calculate the similarity between two articles based on the figure query involves
averaging the scores across all figures in a given article.

**Figure query:** (implemented across FigureGraph.java and ClusterFigure.R)

For each article \( X \) in the set:

- For each figure in article \( X \)
Calculate the related articles (and their corresponding Lucene score) by the figure query (many of the least related articles will likely have a score of 0)

For each article \( Y \) in the union of all articles from the above figure queries

Set the score of article \( Y \) as determined by article \( X \) as the mean score of \( Y \) across all figures in \( X \)

Define a similarity matrix \( X \) where \( X[i, j] = \text{mean of article } j \text{'s score as determined by article } i \text{ and article } i \text{'s score as determined by article } j \) (either of which could be 0)

Note that the similarity matrices produced above are essentially kernel matrices generated by an averaged Lucene similarity score. The notion of distance (i.e., dissimilarity) between articles is implicit in our similarity matrix, however it would be contrived to explicitly calculate this distance for two reasons. First, many articles list a similarity score of 0, which would correspond to an infinite distance. Second, the complexities of Lucene’s scoring algorithm would be difficult to systematically reverse into such a distance.

Luckily we don’t need an explicit distance to cluster our similarity matrices: we can simply apply spectral clustering using the similarity matrix as our kernel matrix [7, 12]. To evaluate the clusterings, we use two metrics. First, we look at the adjusted Rand index between the two clusterings, for various numbers of clusters [4]. This index provides a simple indication of how similar two clusterings are. In essence, this helps us answer the first question we posed in exploring evaluation criteria: how different is our method from conventional tools?

Finally, we also look at average silhouette scores for both clusters. The silhouette of a given article within a clustering is a measure of how close the article is to other articles in its cluster, as well as how far it is from other articles in the nearest neighboring cluster [11]. The average silhouette score for a given clustering is then the average across the silhouette of each of its nodes (articles). Because the standard silhouette score is usually calculated based on distances, and we only have similarities, we simply negate the score; our definition follows:

**Silhouette Score:**

Let \( a(i) \) be the average similarity of article \( i \) with other articles in its cluster

Let \( b(i) \) be the lowest average similarity of article \( i \) with other articles in each cluster that is not its own (\( b(i) \) is thus the average similarity of article \( i \) in its closest neighbor cluster)

Define \( s(i) = \frac{a(i) - b(i)}{\max(a(i), b(i))} \) (for distances, the numerator is \( b(i) - a(i) \))

Calculate the average \( s(i) \) across all articles in the clustering

A silhouette \( s(i) \) close to 1 indicates a datum that is well matched in its own cluster, and that would not fit well in any other cluster. A score of 0 indicates a datum that is on the border of two clusters. And a score of -1 indicates a datum that would fit better in its neighbor cluster. Hence the average silhouette across all data provides a measure of the suitability of a clustering. We calculate such silhouette scores for varying numbers of clusters and plot the results for both the article query and figure query. This allows us to compare the clustering between both methods. We then hypothesize that a tighter clustering could indicate a better relatedness calculation, providing improved ranking of related articles.
In essence, comparing silhouette scores helps us answer the second question we posed in exploring evaluation metrics: does our method improve the ranking of related articles?

Lastly, we also were able to partially explore actual user data from the Elsevier A/B test. From users that used our tool to “hop” from one article to another, we could observe the rank of the latter and see whether users are jumping to the top related articles, or if they’re needing to scroll through several before finding a suitably related article.

4 Results

4.1 Correlation Analysis

The first of our evaluation metrics involved looking at the relationship between the relatedness rankings based on article features alone (the “article query”), and those based on article and figure features together (the “figure query”; see Methods). We started by looking at the correlations between these two methods for 100 randomly selected figures in our corpus. A histogram of these correlations appears in Figure 1, which indicates that the rankings returned by the figure query are positively, but not significantly, correlated with those returned by the article query. We would expect positive correlation since after all the figure query is just a refinement of the article query (and in fact the score assigned by the figure query partially includes the score assigned by the article query). We can also look at our alternative score, which is just the number of overlapping articles in the top 20 of both vectors (Figure 2). Both of these histograms seem to demonstrate that the results of the article query are uniquely altered by the figure query.

Next we applied the top-20 scoring technique to all the figures of several case study articles. One of those case studies is very briefly presented here (I choose not to discuss the case study in detail since evaluating relatedness requires some domain knowledge of the field). For an article on apoptosis in human melanoma cell lines, we looked at the level of overlap between the article query and the results of each figure query (the article has 8 figures). The article can be found here: http://www.sciencedirect.com/science/article/pii/S0196978114000461.

The top 5 related articles by the article query can be found at the following links:


And the results of the figure query for each figure can be found at the following links:

http://imagesearch.sciencedirect.com/#/S0196978114000461/gr1/similar
http://imagesearch.sciencedirect.com/#/S0196978114000461/gr4/similar
http://imagesearch.sciencedirect.com/#/S0196978114000461/gr3/similar
http://imagesearch.sciencedirect.com/#/S0196978114000461/gr2/similar
http://imagesearch.sciencedirect.com/#/S0196978114000461/gr6/similar
4.2 Clustering Analysis

After exploring how the figure query re-ranks the article query, we next clustered several neighborhoods of our corpus by both methods and examined the results. For the same melanoma article discussed above, we used the top 500 related articles by the article query as a neighborhood to examine. As discussed in the Methods section, we clustered these 500 articles using similarity metrics determined by the article query or figure query separately. First we plotted the adjusted Rand index (a measure of cluster similarity) for the two clusterings against the number of clusters specified (Figure 3). This seems to indicate that for any reasonable number of clusters (> 2), the two clusterings are relatively different (low adjusted Rand score).

Next we examined the average silhouette score for both clusterings for various specified
Figure 2: Number of overlapping articles in the top 20 results of the article and figure queries for 100 randomly selected figures.

We also repeated our clustering analysis for the neighborhood of another article of interest (http://www.sciencedirect.com/science/article/pii/S0143417905001344). The article focuses on mRNA distribution in mouse brains. The same plots appear in Figures 5 and 6 and indicate very similar results.

Finally, we also briefly looked at user data from the Elsevier A/B test. From this data, we can see when a user “hops” from one article to another. More importantly, we can also observe the rank of the article that they jumped to from the article they left. This allows us to see if users are jumping to the top related articles, or if they are needing to scroll through several to reach a suitably related article. The mean ranking of such jumps was 9.9. This is somewhat high, though our data set is still very small (around 60 such “hops” are documented); as scientists continue to use our tool, our data set will grow and we can get a better indicator of how suitable our top related documents are.
Discussion

Using only textual contexts so far, we have built a decently sophisticated related images feature on top of the Yale Image Finder. Such a feature should prove useful to researchers and medical practitioners alike in accessing experiments and techniques; cross-validating results; and performing literature reviews.

My work this term focused on caching our related image results for rapid access, and exploring various metrics to evaluate our novel relatedness method. The caching of results was successfully completed in 4 days with parallelization across 16 cores.

In exploring metrics of evaluation, we hoped to answer two (albeit related) questions: 1) How different is our method from conventional methods? and 2) Does our method return superior results to conventional methods? To explore the first of these questions, we looked at correlations and overlap between the results of our full method (utilizing figure and article features) and those of only the first part of our method (utilizing solely article features). The results indicate that incorporating figure information does actually re-rank relatedness results.

We subsequently implemented spectral clustering analysis based on the similarities defined by both methods. The spectral clustering method was chosen for its robustness and convenience as a method that can be applied directly to a similarity matrix (as opposed to a distance matrix). Using an adjusted Rand index, we compared how similarly the two methods clustered a given set of articles; for any clustering with more than 2 clusters, the two methods resulted in significantly different groupings. Finally, we used an average silhou-
Figure 4: Average silhouette score for article query and figure query clusterings, for various numbers of clusters, and for the neighborhood of articles around one on melanoma.

... ette score to evaluate the tightness of our clusterings. The results indicated that clustering based on article and figure features together resulted in higher silhouette scores than those based on article features alone. It is important to note that a higher silhouette score does not necessarily imply a superior relatedness ranking; after all, it is possible that several methods would return the same ranking while clustering somewhat differently. However, we hypothesize that a tighter clustering could indicate better resolution of the underlying interrelatedness between articles, which would result in superior rankings.
Figure 5: Adjusted Rand index between article query and figure query clusterings, for various numbers of clusters, and for the neighborhood of articles around one on mouse brains.

6 Supplementary Materials

6.1 Prior Methods

Another feature we wanted to implement as part of the related images tool was the option to restrict results to a specific image category (diagram, “real”, or chart), the motivation being that a user may already know what type of image she is searching for when using our tool. As such, we built a trained image classifier. Beginning with a training set of 10,000 images annotated by Mate Nagy, we examined each figure and counted the number of appearances of each term in the figure’s caption (i.e., we calculated the term frequency); next, the term count for the terms in each category was divided by the corresponding count of the same term in the other two categories (i.e., multiplied by the inverse document frequency), yielding a \( \text{tf} \times \text{idf} \) score for each term by class (see LearnTypes.java and ProcessClassifiedTerms.java). We can subsequently search for images from the same class by including the top \( \text{tf} \times \text{idf} \) terms for that class. The way this works is as follows: given a source image, we attempt to identify it as belonging to one of the three classes by checking the alignment of the caption with the top terms for each class – strong intersection would indicate that an image is of a particular class. If we can strongly identify an image as a given type, we subsequently include a query composed of the top 200 terms for that class (each weighted by \( \text{tf} \times \text{idf} \) score) in our overall query for similar images. This additional query should result in a greater score for images of the same class, thus refining our search. (Note that terms were properly stemmed using Porter’s stemming algorithm in Lucene while building the classifier.)
Figure 6: Average silhouette score for article query and figure query clusterings, for various numbers of clusters, and for the neighborhood of articles around one on mouse brains.

6.2 Prior Results

We examined the following six figures to test our feature: S0143417905001344, S1074742709000677, S1074742709000677, S0031938411005622, S0031938411005622, S0166432813002878. In the following segment, I discuss the improvements we observed between the original, proof-of-concept feature and the most recent iteration of our tool. I only describe one of the six case studies here, since the improvements are similar across the board and begin to involve nuances of the various biomedical fields presented.

Figure 7 shows the first figure listed above, along with its caption. The image represents various mouse brain samples stained for a certain molecule. In the original related images implementation – in which only image captions were used for determining relatedness – the feature returned images of mouse brains, but with different stains; Figure 8 shows the top result from the original feature. Looking at the source image’s caption (Figure 7), we can see that it contains an array of acronyms corresponding to brain regions; thus any image captions matching these regions result in top hits (such as that in figure 8). Simply returning images of mouse brains is not very useful, however – what we most likely want are images of mouse brains stained for the same molecule.

In our most recent advance on the related images feature, we get far superior results; Figure 9 shows the top hit from the new implementation. Examining this figure’s caption, we can see that it also includes the list of acronyms that the source image includes. Yet the image still displays the proper stain we want – this is almost certainly a result of our two-part strategy, which searches for related articles before retrieving related images from
that subset. In filtering our search by the set of related articles, we ensure that the figures we retrieve match the topic (in this case the molecule being stained for), rather than just the general details of a specific experiment (in this case simple brain regions).

In an attempt to optimize the so-called context radius (the number of articles returned in the first stage of our two-part method), we plotted the top relatedness scores in decreasing order. A greater radius may include more potentially important images, but it may also result in loss of specificity. Most importantly, a greater context radius increases the number of images to query, resulting in significant increases in processing time. From the case studies described above, we learned that a radius of 50 articles was too small to capture some of the top hits we discovered, while a radius of 1000 significantly increased the load time (by about 5 seconds). Figure 10 displays the (log) relatedness scores of the top 1000 hits for four of the case studies above. Observing a slight drop-off around 300-400 articles for some of the figures, and not noticing significant improvements after 300 articles, we tentatively used 300 as the parameter for the above cases. Later we chose 500 as a decent radius.

The last set of results we gathered this term come from the image classifier. As described in the Methods section, the classifier returned the top 200 terms for each of the three classes defined, based on a manually implemented tf*idf statistic. The top 40 of these terms are presented in Figure 11 with their associated tf*idf score. Observing the terms, some of them clearly make sense: “diagrammat”, “taxonomi”, and “framework” appear for diagrams; “anteroposterior” (in relation to a body scan of some kind), “chest”, and “abdomen” appear under real terms; and “predictor” and “quintil” appear for charts. However a good number of terms do not intuitively relate to these three classes. Examining these odd terms more closely (by printing the full caption), we concluded that the statistics are in fact accurate; these terms just appear in our training set for certain classes. Furthermore, some of these top tf*idf terms may be indicative of a given class, but they certainly bias the results towards a particular subject area. For example, “smoker” appears with a high tf*idf stat for charts, presumably because a lot of medical papers examined and subsequently plotted effects of smoking. However utilizing such a top term to refine the search towards other charts would simultaneously favor figures involving smoking, when the chart in question may have nothing to do with smoking. Because of the lukewarm results, we tabled the classifier for now – once we implement texture analysis, however, we should be able to achieve much greater classification (since a chart and an x-ray, for example, have very different textures).

6.3 Prior Discussion

The original version of this tool, written prior to my CS 490 project, queried related figures using image captions. While this implementation provided rudimentary relatedness, my goal in Fall 2014 was to build the feature out. This process involved (chronologically): acquainting myself with Java and Lucene; building a platform to test changes to the tool; iteratively modifying the queries for relatedness; designing a classifier to supplement related queries by type; and finally separating article and figure queries for a two-part approach. The most recent version appears to offer a significantly higher caliber of similarity than the original proof-of-concept implementation, as determined by various case studies. While the ternary classifier we built did not appear to substantially affect results from the cases we examined, there are interesting ways we could continue to improve upon the model using
Figure 7: Source figure for related images; caption from paper: “Autoradiograms of prodynorphin mRNA expression in the mouse. The slides containing coronal mouse brain sections are presented in rostrocaudal sequence. Some major neuroanatomical landmarks are indicated on the right side of each autoradiogram. Abbreviations: AO, anterior olfactory nucleus; AcbC, nucleus accumbens; Arc, accurate hypothalamic nucleus; Ce, central amygdaloid nucleus; Cpu, caudate/putamen; DG, dentate gyrus; DMH, dorsomedial hypothalamic nucleus; LEnt, lateral entorhinal cortex; LH, lateral hypothalamic area; L Rt, lateral reticular nucleus; NTS, nucleus of the solitary tract; Pir, piriform cortex; PVN, paraventricular hypothalamic nucleus; RSA, retrosplential cortex; SO, supraoptic nucleus; Tu, olfactory tubercle; VMH, ventromedial hypothalamic nucleus. Magnification, 10X.” [10]
Figure 8: Result from original implementation; caption from paper: “In situ hybridization shows regional expression of Gpr101 in brain. In situ hybridization images of coronal (A, B, D, E, GS, UX) or sagittal (C, F) mouse brain sections or spinal cord (T) exposed to photographic emulsion. Images are dark field except in panel A where dark- and bright-field images are overlaid to show details of the tissue. Probe specificity is demonstrated in UX where adjacent sections are hybridized with sense or anti-sense probes as indicated. Microscopic images obtained using a 4X (C, D, H, JN, P, R, and T) or 10X (A, B, E, FG, I, O, Q, S, UX) objective. Scale bars indicate 100 um. Film autoradiograms of representative coronal sections are shown in Y, Z and AA. Abbreviations 3V, 3rd ventricle; 4V, 4th ventricle; 7n, facial nerve; aca, anterior commissure anterior part; AcbSh, accumbens nucleus shell; AHA, anterior hypothalamic area; AHi, amygdalohippocampal area; AP, area postrema; APir, amygdalopiriform transition area; Aq, aqueduct (Sylvius); Arc, arcuate hypothalamic nucleus; BST, bed nucleus of the stria terminalis; CC, central canal; Ce, central amygdaloid nucleus; CGPn, central gray of the pons; DM, dorsomedial hypothalamic nucleus; DMPAG, dorsomedial periaqueductal gray; Ent, entorhinal cortex; f, fornix; Gl, glomerular layer of the olfactory bulb; GrO, granular layer of the olfactory bulb; ic, internal capsule; IGL, intergeniculate leaf; IRt, intermediate reticular nucleus; LH, lateral hypothalamic area; LM, lateral mammillary nucleus; LSI, lateral septal nucleus intermediate part; LSV, lateral septal nucleus ventral part; LV, lateral ventricle; MePD, medial amygdaloid nucleus posterodorsal part; MePV, medial amygdaloid nucleus posteroverentral part; Mi, mitral cell layer of the olfactory bulb; ML, medial mammillary nucleus lateral part; MM, medial mammillary nucleus medial part; MPA, medial preoptic area; MPO, medial preoptic nucleus; mt, mammillothalamic tract; opt, optic tract; Pa, paraventricular hypothalamic nucleus; PeF, perifornical nucleus; PH, posterior hypothalamic area; PnC, pontine reticular nucleus caudal part; Pr, prepositus nucleus; Py, pyramidal cell layer of the hippocampus; S, subiculum; SCh, suprachiasmatic nucleus; SFO, subfornical organ; SO, supraoptic nucleus; SolM, nucleus of the solitary tract medial part; VLG, ventral lateral geniculate nucleus; VLPAG, ventrolateral periaqueductal gray; VMH, ventromedial hypothalamic nucleus.”
Figure 9: Result from improved implementation; caption from paper: “Low-magnification film autoradiographs show the location of NPY mRNA expression (A, B and C). The maps of A’, B’ and C’ are adopted from a rat brain atlas (Paxinos and Watson, 1997) indicating where the levels of the mRNA expression were measured. Abbreviations: Acb, accumbens nucleus; AI, agranular insular cortex; Amg, amygdaloid nucleus; Arc, arcuate hypothalamic nucleus; Au, auditory cortex; CA, fields CA of Ammon’s horn; ACC, anterior cingulate cortex; CPu, caudate putamen; DG, dentate gyrus; DMC, dorsomedial hypothalamic nucleus; compact part, LSV, lateral septal nucleus; ventral part, MePD, posterodorsal part of the medial amygdaloid nucleus; Pir, piriform cortex; PMCo, posteromedial cortical amygdaloid nucleus; RSG, retrosplenial granular cortex; Rt, reticular thalamic nucleus; Tha, thalamus; Tu, olfactory tubercle nucleus; Vis, visual cortex; VMH, ventromedial hypothalamic nucleus.” [6]
Figure 10: Relatedness scores for images similar to three of the six case study images.
### TOP DIAGRAM TERMS

- raxml 100
- angiotensin 100
- unroot 100
- ellipsoid 19.33333333
- concept 19
- stm 9
- titl 8.166666667
- succinyl 8
- diagrammat 8
- hydrogenas 8
- sulfur 7
- sav 7
- pocket 6.4
- machineri 6
- paracrin 6
- vertex 6
- skew 6
- prefix 6
- stereoview 6
- acp 6
- atom 5.6
- mega 5.5
- taxonomi 5
- pharmaceut 5
- symmetri 5
- capsid 5
- dihydrofol 5
- crystallograph 5
- pack 5
- kimura 5
- catalys 5
- intramolecular 5
- lsa 5
- aldolas 5
- stick 4.875
- framework 4.666666667
- conceptu 4.5
- parsimoni 4.5
- pymol 4.5
- bootstrap 4.222727272

### TOP REAL TERMS

- habitu 12
- conjunctiva 10
- pulp 10
- pat 10
- intima 9
- gu 9
- ciliari 9
- uteru 8
- jar 7
- dsi 7
- bhi 7
- anteroposterior 6
- cathet 6
- cuticl 6
- rex 6
- desmin 6
- foreign 6
- ruffl 6
- toluidin 6
- clc 5.5
- john 5.5
- opl 5.25
- flair 5
- william 5
- abdomen 4.5
- nup 4.5
- chest 4.333333333
- itch 4.333333333
- lime 4.333333333
- punctat 4.333333333
- arch 4.25
- escc 4
- thicken 4
- mediastin 4
- patenc 4
- dctp 4
- anilin 4
- autoimmun 4

### TOP CHART TERMS

- zeta 100
- xylos 100
- hematocrit 100
- imput 100
- smoker 100
- urin 100
- cultivar 100
- hazard 17
- hco 14
- haemoglobin 14
- predictor 13
- dunnett 13
- fece 13
- sap 12
- wear 10
- law 10
- forag 10
- δh 10
- systol 9.666666667
- bland 9
- asthma 9
- gametocytc 9
- ppar 9
- δδct 9
- deactiv 9
- vt 8
- fpr 8
- coral 8
- sputum 8
- lytic 8
- mca 8
- locomotor 8
- adsorpt 8
- fmirl 8
- quintil 8
- intent 8
- ketoglutar 7.666666667
- entropi 7.5
- oscillatori 7

---

**Figure 11:** Top terms with tf*idf scores for three defined categories of images.
feature analysis techniques.

Finally, in our two-stage relatedness search process, we would ideally like to recalculate tf*idf stats using this subspace as the corpus for “document frequency”. While this should not change the results drastically, it may change the order of our results in some interesting ways. Unfortunately the only way to do this currently seems to be to construct a Lucene index on the fly, which is somewhat impractical.
References


[10] Lin et. al., *Distribution of prodynorphin mRNA and its interaction with the NPY system in the mouse brain*. Neuropeptides, 2006

