

Visual Navigation of Digital Libraries: Retrieval and Classification of Images in the National Library of Norway's Digitised Book Collection

Supplementary materials

1. Full-text search details

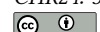
To allow image searches based on surrounding text, our API needs to support full-text search alongside embedding-based image search. Full-text search typically work by computing a relevance score between the search query and the searchable texts. This can be done by first tokenising the query string and computing relevancy scores for each token for each document, before summing these to obtain a relevancy score for each document. A common way to compute the token relevancy score is by multiplying the term frequency (TF) (i.e. how often a token occurs in a specific document) and the inverse document frequency (IDF) (i.e. how seldomly a token occurs any document) and multiplying these to compute the TF-IDF [1]. The key insight to making full-text search efficient is that the TF and IDF can be pre-computed for all tokens in the database. Then, at query-time, an inverted index is used to efficiently obtain the TF for the subset of documents that contain the search tokens.

Although Qdrant lacks built-in full-text search, it is possible to implement it by leveraging its sparse vector capabilities. We used TF-IDF ranking (as described in [2]). Specifically, we tokenised the textual context of each image and computed their TF vector based on token frequencies and the global IDF vector. When a string query is received, it is tokenised, and the corresponding IDF vector entries are queried to form a sparse vector with nonzero IDF values for search tokens. Finally, a maximum inner product search retrieves the most relevant image.

2. Computational demands

The database, consisting of information on 422589 graphical elements from books published before the year 1900 requires a total of 3.1 GiB storage – including all image embeddings, the inverse document frequency vectors and book metadata (e.g. Author). The app is efficient – on average¹, searching based on text takes 0.3 seconds, based on image ID (i.e. using pre-computed embeddings) takes 0.3 seconds and image search with user-uploaded images takes 3 seconds. While image search with user-uploaded images are time-consuming, the bulk of this time is

CHR24: 5th Conference on Computational Humanities Research, December 04–06, 2024, Aarhus, Denmark

 © 2024 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

¹Based on 200 queries

not the query itself, but uploading the image and pre-processing it on the Google Cloud Run instance.

3. Nested cross-validation

Cross-validation [3] can be used for model-selection and model-evaluation, and works by splitting a dataset into K_{inner} -parts, or *folds*. If used for model selection, we fit candidate models using $K_{\text{inner}} - 1$ folds, evaluating on the K_{inner} th fold. We repeat this process K times using each fold for training $K_{\text{inner}} - 1$ times and evaluation once and by averaging the evaluation performance across all K_{inner} folds, we can evaluate which model-candidate performed the best. Using the best model-candidate, we train it again using the full dataset to obtain a model for use on new, unseen, data. This is described in Algorithm 1

However, if we're using cross-validation for model-selection, then we will not have accurate estimates for performance: we're testing many models, and we are likely to have selected a model which was better by random chance [4]. To counteract this, it is common to use a separate hold-out test set that is used for a final evaluation. However, a more powerful approach is to use another cross-validation loop, splitting the dataset into K_{outer} folds, using cross validation to select and fit a model based on $K_{\text{outer}} - 1$ folds and evaluating only the selected model on the final fold. Finally, after model performance is computed on all K_{outer} folds, we use cross-validation based on K_{inner} folds to select and fit a model. This is called *nested cross-validation* and the full procedure is described in Algorithm 2. Nested cross-validation can be time-consuming, requiring fitting $O(N_{\text{models}} K_{\text{inner}} K_{\text{outer}})$ models in total.

References

- [1] K. Spärck Jones, A Statistical Interpretation of Term Specificity and its Application in Retrieval, *Journal of Documentation* 28 (1972) 11–21. doi:10.1108/eb026526.
- [2] D. Turnbull, J. Berryman, *Relevant Search: With Applications for Solr and Elasticsearch*, Manning Publications Co, Shelter Island, New York, 2016.
- [3] T. Hastie, R. Tibshirani, J. H. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Springer Series in Statistics, 2nd ed ed., Springer, New York, NY, 2009.
- [4] C. Ambrose, G. J. McLachlan, Selection bias in gene extraction on the basis of microarray gene-expression data, *Proceedings of the National Academy of Sciences* 99 (2002) 6562–6566. doi:10.1073/pnas.102102699.

Algorithm 1: K-fold cross-validation for selecting model parameters

Data:

Collection image-label pairs: $\mathcal{X} = \{(\mathbf{i}_1, y_1), \dots, (\mathbf{i}_N, y_N)\}$;

Collection of regularisation parameters: $\mathcal{C} = \{C_1, \dots, C_S\}$;

Collection of functions so that $f_p(\mathbf{i}_i)$ is the embedding vector for image \mathbf{i}_i : $\{f_1, \dots, f_P\}$;

The number of inner cross-validation folds: K_{inner} ;

Result:

Optimal model complexity, \hat{C} , and preprocessing \hat{f} ;

Fitted model ;

begin

Partition the dataset, \mathcal{X} into K_{inner} equally sized parts: $\mathcal{X}_1, \dots, \mathcal{X}_{K_{\text{inner}}}$;

Create sub-datasets, $\mathcal{X}_1, \dots, \mathcal{X}_{K_{\text{inner}}}$, where \mathcal{X}_{-i} consists of all images in \mathcal{X} that are not in \mathcal{X}_i (i.e. $\mathcal{X}_{-i} \cap \mathcal{X}_i = \emptyset$ and $\mathcal{X}_{-i} \cup \mathcal{X}_i = \mathcal{X}$) ;

for $i = 1 \dots K_{\text{inner}}$ **do**

for Each combination of C_s and f_p **do**

 Compute image embeddings $\mathbf{x}_i^{(p)} = f_p(\mathbf{i}_i)$ for all images in \mathcal{X} ;

 Fit a ridge regularised logistic regression model, $g(\mathbf{i}_i; C_s, f_p) \approx y_i$ using \mathcal{X}_{-i} as the training set. Use f_p as a preprocessing function for the images and set the complexity parameter (inverse regularisation parameter) to C_s ;

 Compute the F1 score for $g(\mathbf{i}_i; C_s, f_p)$ on \mathcal{X}_i ;

end

end

Select the regularisation paramter, $\hat{C} = C_s$, and preprocessing function, $\hat{f} = f_p$, that obtained the highest F1 score across all K_{inner} folds. ;

Fit a logistic regression model, $g(\mathbf{i}; \hat{C}, \hat{f})$ using \mathcal{X} as the training set ;

end

Algorithm 2: Nested cross-validation for estimating model performance and model parameters

Data:

Collection image-label pairs: $\mathcal{X} = \{(\mathbf{i}_1, y_1), \dots, (\mathbf{i}_N, y_N)\}$;

Collection of regularisation parameters: $\mathcal{C} = \{C_1, \dots, C_S\}$;

Collection of functions so that $f_p(\mathbf{i}_i)$ is the embedding vector for image \mathbf{i}_i : $\{f_1, \dots, f_P\}$;

The number of outer and inner cross-validation folds: K_{outer} and K_{inner} ;

Result:

Model performance estimate ;

Optimal regularisation parameter, \hat{C} ;

Fitted model ;

begin

 Partition the dataset, \mathcal{X} into K_{outer} equally sized parts: $\mathcal{X}_1, \dots, \mathcal{X}_{K_{\text{outer}}}$;

 Create sub-datasets, $\mathcal{X}_1, \dots, \mathcal{X}_{K_{\text{outer}}}$, where \mathcal{X}_{-i} consists of all images in \mathcal{X} that are not in \mathcal{X}_i (i.e. $\mathcal{X}_{-i} \cap \mathcal{X}_i = \emptyset$ and $\mathcal{X}_{-i} \cup \mathcal{X}_i = \mathcal{X}$) ;

for $i = 1 \dots K_{\text{outer}}$ **do**

 Use K-fold cross-validation (Algorithm 1) to fit a logistic regression model,

$g_i(\mathbf{i}; \hat{C}_i, \hat{f}_i)$, with optimal parameters \hat{C}_i and \hat{f}_i based on \mathcal{X}_{-i} . ;

 Compute evaluation metrics on \mathcal{X}_i ;

end

 Use K-fold cross-validation (Algorithm 1) to fit a logistic regression model, $g(\mathbf{i}; C, f)$, with optimal parameters C and f based on \mathcal{X} . ;

end
