Visualization of Large Text Collections in 2D

Stjepan Antivo Ivica, Frano Perleta

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
{frano.perleta,stjepan-antivo.ivica}@fer.hr

Abstract

Large collections of text may be vizualized through dimensionality reduction while preserving semantic similarity between documents. We implement both Latent Semantic Analysis and Multidimensional Scaling, and apply those techniques to a collection of newsgroup messages.

1. Introduction

When faced with large amounts of data of any kind, it is often useful to represent it visually in order to leverage the innate human ability to recognize patterns and spatial relationships. This involves mapping the data into 2D or 3D space, in a way which preserves information meaningful for a particular application.

The goal of our project is visualization of large collections of textual documents. The primary objective is to capture the semantics of text in some way so that the visual representation of the collection reflects relationships between documents such as pertinence to related topics. There are many ways to extract semantic information from natural language, with various tradeoffs in terms of precision, computational feasibility, etc.

Our assignment is to apply the two methods of Latent Semantic Analysis and Multidimensional Scaling (henceforth refered to as LSA and MDS respectively) to this problem. We assume a very simple language model, representing documents with elements of a high-dimensional vector space. The problem can therefore be formulated as dimensionality reduction, and in principle any of the various techniques may be used, not necessarily specific to natural language processing.

Loosely speaking, the degree to which any two particular documents overlap in topic, ie. their semantic similarity, induces a certain topology on the document space. While this is a very informal idea, LSA and MDS are techniques intended to approximate that topology in a low-dimensional space.

We apply our implementation to the 20-newsgroups dataset[1], which is a large collection of newsgroup messages. By their very nature, individual messages are classified into 20 newsgroups, which are themselves grouped into 6 coarse classes. One would expect documents within a class to be close to each other, but there might also be considerable overlap between classes.

2. Related research

Large-scale Latent Semantic Indexing is discussed at length in (Zhang and Zhu, 2008). Although not directly related to visualization, the algorithm described there directly inspired our implementation of LSA.

[1] http://qwone.com/~jason/20Newsgroups/

3. Description

Our implementation consists of multiple standalone programs:

- `tar-lsa` is a command-line tool written in Haskell which performs dataset preprocessing and transformation using LSA.
- `mds.jar` is a command-line tool written in Java. It transforms a collection of vectors using MDS.
- `visual.jar` provides an interactive graphical interface which enables users to explore a 2D visualization of the dataset and retrieve individual documents.

These programs require that the dataset is available in two forms: as a collection of text files, each containing a single message; and as MATLAB®-compatible matrix files, containing among other things the word counts for all documents. Both versions of the dataset are available online.

3.1. LSA preprocessing

Due to the size of the dataset (19k documents and 61k terms), it is impractical, if not outright infeasible, to store the entire term-document matrix and calculate its singular value decomposition. The `tar-lsa` tool is used to determine a subset of documents and terms which will be used to construct a close approximation to the real term-document matrix.

The dataset is processed as a stream of document vectors, each thrown away immediately after updating the accumulators. When the entire pass through the dataset is complete, the magnitude of each document vector and the total number of occurences of each term are known, as well as the number of distinct documents containing each term.

- A fixed number of documents is selected, in order of decreasing ℓ^2-norm. This minimizes the approximation error with respect to the Frobenius norm (Zhang and Zhu, 2008).
- Terms are left out if they have too few occurences, since they are likely noise; or if they occur in too few distinct documents, since such terms contain very little "topological" information we are interested in.

- The identifiers of selected documents and terms are written to a pair of files to be read by subsequent invocations of `tar-lsa lsa`.

All the parameters for selection may be passed via command-line arguments – detailed instructions are available by invoking `tar-lsa prepare --help`.

3.2. LSA transformation

The transformation needs to be computed only once for a particular choice of output dimensionality. It proceeds as follows:

1. The identifiers of N documents and M terms which were previously selected are loaded from the filesystem. A dense $M \times N$ matrix $\mathbf{A}$ is allocated, and a pair of `IntMaps` is used to map sparse document and term identifiers into contiguous matrix indices.

2. A pass is made through the dataset, and selected documents are used to initialize the matrix $\mathbf{A}$. Subsequently, tf-idf weighting is applied to $\mathbf{A}$, with augmented term frequencies:

$$tf(t,d) = \frac{1}{2} + \frac{\mathbf{A}_{t,d}}{2 \max_\tau \mathbf{A}_{\tau,d}} \tag{1}$$

3. The thin singular value decomposition of $\mathbf{A}$ is computed:

$$\mathbf{A} = \mathbf{U}_k \mathbf{\Sigma}_k \mathbf{V}_k^\top, \tag{2}$$

   where $k = \min\{M, N\}$ (which is N in practice), $\mathbf{U}_k$ is $M \times k$, $\mathbf{\Sigma}_k$ is $k \times k$, and $\mathbf{V}_k$ is $N \times k$. When seen as a linear operator, $\mathbf{A}$ is effectively factorized through a k-dimensional concept space. The columns of $\mathbf{U}_k$ represent concepts as basis vectors in the document space, while the numbers on the main diagonal of $\mathbf{\Sigma}_k$ signify the relative importance of each concept.

4. Another pass is made, in which each individual document vector (consisting of occurence counts) is first orthogonally projected onto the M-dimensional subspace, normalized, and finally mapped into the concept space:

$$\hat{\mathbf{d}} = \mathbf{\Sigma}_k^{-1} \mathbf{U}_k^\top \mathbf{d} \tag{3}$$

   If the output dimensionality is D, the final orthogonal projection is onto the subspace spanned by the vectors corresponding to D largest singular values. The output vectors are written to the filesystem.

The most computationally intensive part of the transformation is the SVD. We used the `hmatrix` library (Ruiz, 2014) for Haskell, which provides a purely functional interface to BLAS, LAPACK and GSL. The entire transformation takes on the order of tens of seconds, depending on M and N. The *thin* SVD algorithm provided by the `hmatrix` library avoids the costly and unnecessary calculation of singular vectors not corresponding to the largest k singular values.

For detailed usage instructions, run `tar-lsa lsa --help`.

3.3. MDS transformation

Whereas LSA uses a low-dimensional concept space to capture semantic similarity, MDS uses a dissimilarity measure on the document space and produces a mapping into a low-dimensional space which preserves that measure as well as possible. This method, as well as issues researches are confronted with when applying it, are discussed in (Wickelmaier, 2003). We used MDSJ, a free Java library which implements the algorithm described in (Brandes and Pich, 2007).

The computation of MDS proved to be memory consuming. To alleviate this problem, we first pass the document vectors through LSA in order to reduce the dimensionality to k. We experimented with various settings such as $k = 4$, $k = 1000$. The best way to calculate dissimilarity is Manhattan distance due to high similarity among the documents. Processing is done in batches of 200 documents to further scatter their representations in the final space.

3.4. Visualization

The visualization tool `visual.jar` is built in Java using the Prefuse library (Heer et al., 2005). Its arguments are: the path to a file containing triples of the form $(documentID, x, y)$; the path to a file containing numeric labels of each document; a flag that switches between fine or coarse classes; and path to the directory containing raw text documents. The graphical interface is interactive. The user can navigate using left clicks, zoom in and out using right clicks, and reset the view using double right clicks.

All documents are represented with shapes determined by coarse classes. Fine classes are distinguished using color. Information about a document is displayed when the user hovers the mouse cursor over the shape. Left clicking on the shape opens the corresponding text file in the directory provided as the last command-line argument.

4. Evaluation

Different visualizations of a subset of the document collection can be seen in figure 1.

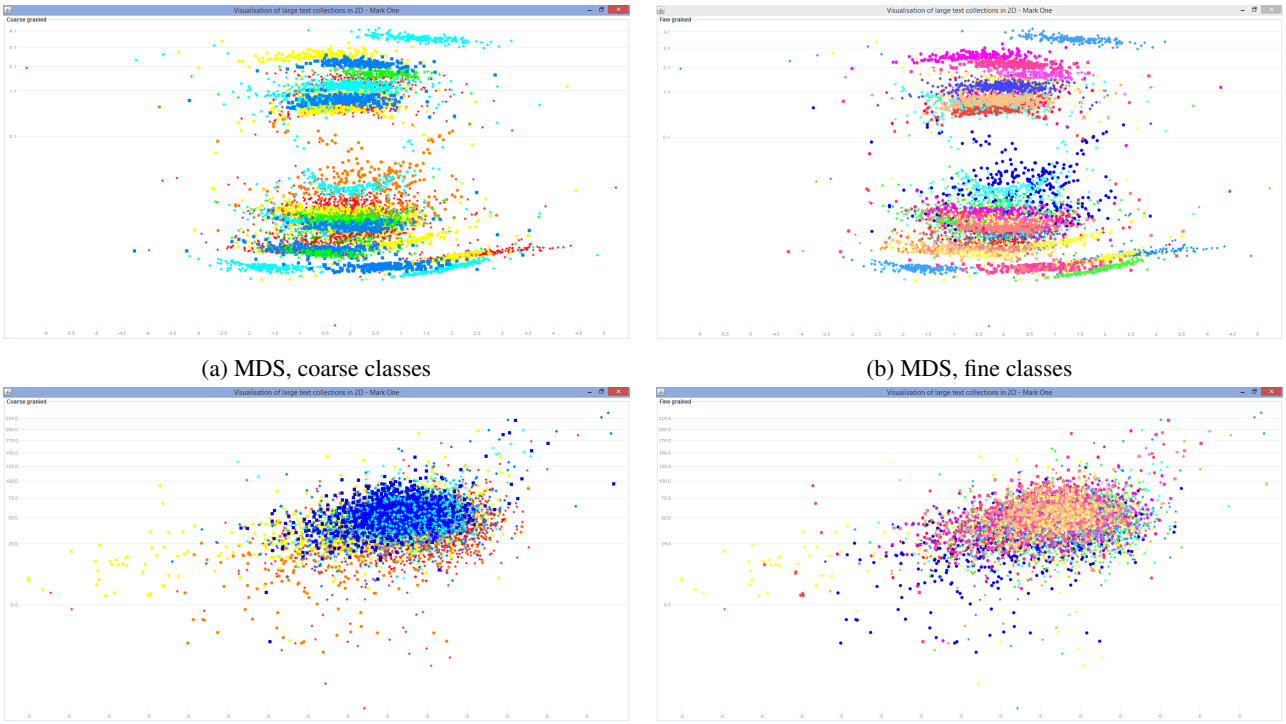
LSA by itself seems insufficient for this purpose. Although not entirely useless, the classes overlap almost completely, and almost no insight can be gained through visual inspection.

MDS yields somewhat more satisfying results. Although the contiguity of coarse classes has been lost, it is mostly preserved for fine classes.

5. Conclusion

LSA and MDS are two relatively simple and well understood techniques applicable to text visualization. Both can be calculated efficiently. However, our experiments with these ended with mediocre results.

Although we are certain that a more rigorous implementation of these techniques would likely surpass ours in efficiency and usefulness, it seems worthwhile to explore other possible approaches to text collection visualization, such as deep learning.



(a) MDS, coarse classes

(b) MDS, fine classes

(c) LSA, coarse classes

(d) LSA, fine classes

Figure 1: Visualization of a subset of the 20-newsgroups dataset with both algorithms and classification schemes.

References

Ulrik Brandes and Christian Pich. 2007. Eigensolver methods for progressive multidimensional scaling of large data. In *Graph Drawing*, pages 42–53. Springer.

Jeffrey Heer, Stuart K. Card, and James Landay. 2005. Prefuse: A toolkit for interactive information visualization. In *ACM Human Factors in Computing Systems (CHI)*, pages 421–430.

Alberto Ruiz. 2014. hmatrix: A haskell library for numerical computation. `https://github.com/albertoruiz/hmatrix`.

F. Wickelmaier. 2003. An introduction to MDS. *Reports from the Sound Quality Research Unit (SQRU)*, 7.

Dell Zhang and Zheng Zhu. 2008. A fast approximate algorithm for large-scale latent semantic indexing. In *Digital Information Management, 2008. ICDIM 2008. Third International Conference on*, pages 626–631. IEEE.