Latent semantic analysis

Semantics

Language · Linguistics

- Formal semantics (logic & linguistics)
- Lexis
  - Lexical semantics
  - Statistical semantics
  - Structural semantics
  - Prototype semantics
- Lexicology
- Semantic analysis
- Latent semantic analysis
- Theory of descriptions
- Force dynamics
- Unsolved problems
- Semantic matching
- Analysis (machine)
- Abstract semantic graph
- Semantic Web
- Semantic wiki
- Semantic file system
- Abstract interpretation
- Formal semantics of programming languages
  - Denotational semantics
  - Axiomatic semantics
  - Operational semantics
  - Action semantics
  - Categorical semantics
  - Concurrency semantics
  - Game semantics
- Predicate transformer

• v
• t
• e [1]

Latent semantic analysis (**LSA**) is a technique in natural language processing, in particular in vectorial semantics, of analyzing relationships between a set of documents and the terms they contain by producing a set of concepts related to the documents and terms. LSA assumes that words that are close in meaning will occur in similar pieces of text. A matrix containing word counts per paragraph (rows represent unique words and columns represent each paragraph) is constructed from a large piece of text and a mathematical technique called singular value decomposition (SVD) is used to reduce the number of columns while preserving the similarity structure among rows. Words are then compared by taking the cosine of the angle between the two vectors formed by any two rows. Values close to 1 represent very similar words while values close to 0 represent very dissimilar words.

LSA was patented in 1988 (US Patent 4,839,853 [2]) by Scott Deerwester, Susan Dumais, George Furnas, Richard Harshman, Thomas Landauer, Karen Lochbaum and Lynn Streeter. In the context of its application to information retrieval, it is sometimes called Latent Semantic Indexing **(LSI)**.

Overview

Occurrence matrix

LSA can use a term-document matrix which describes the occurrences of terms in documents; it is a sparse matrix whose rows correspond to terms and whose columns correspond to documents. A typical example of the weighting of the elements of the matrix is tf-idf (term frequency–inverse document frequency): the element of the matrix is proportional to the number of times the terms appear in each document, where rare terms are upweighted to reflect their relative importance.

This matrix is also common to standard semantic models, though it is not necessarily explicitly expressed as a matrix, since the mathematical properties of matrices are not always used.

Rank lowering

After the construction of the occurrence matrix, LSA finds a low-rank approximation[3] to the term-document matrix. There could be various reasons for these approximations:

- The original term-document matrix is presumed too large for the computing resources; in this case, the approximated low rank matrix is interpreted as an *approximation* (a "least and necessary evil").
- The original term-document matrix is presumed *noisy*: for example, anecdotal instances of terms are to be eliminated. From this point of view, the approximated matrix is interpreted as a *de-noisified matrix* (a better matrix than the original).
- The original term-document matrix is presumed overly sparse relative to the "true" term-document matrix. That is, the original matrix lists only the words actually *in* each document, whereas we might be interested in all words *related to* each document—generally a much larger set due to synonymy.

The consequence of the rank lowering is that some dimensions are combined and depend on more than one term:

{(car), (truck), (flower)} --> {(1.3452 * car + 0.2828 * truck), (flower)}

This mitigates the problem of identifying synonymy, as the rank lowering is expected to merge the dimensions associated with terms that have similar meanings. It also mitigates the problem with polysemy, since components of polysemous words that point in the "right" direction are added to the components of words that share a similar meaning. Conversely, components that point in other directions tend to either simply cancel out, or, at worst, to be smaller than components in the directions corresponding to the intended sense.

Derivation

Let X be a matrix where element (i, j) describes the occurrence of term i in document j (this can be, for example, the frequency). X will look like this:

$$\begin{array}{c} \\ \mathbf{t}^T \rightarrow \end{array} \begin{array}{c} \mathbf{d}_j \\ \downarrow \\ \begin{bmatrix} x_{1,1} & \dots & x_{1,n} \\ \vdots & \ddots & \vdots \\ x_{m,1} & \dots & x_{m,n} \end{bmatrix} \end{array}$$

Now a row in this matrix will be a vector corresponding to a term, giving its relation to each document:

$$\mathbf{t}_i^T = \begin{bmatrix} x_{i,1} & \dots & x_{i,n} \end{bmatrix}$$

Likewise, a column in this matrix will be a vector corresponding to a document, giving its relation to each term:

$$\mathbf{d}_j = \begin{bmatrix} x_{1,j} \\ \vdots \\ x_{m,j} \end{bmatrix}$$

Now the dot product $\mathbf{t}_i^T \mathbf{t}_p$ between two term vectors gives the correlation between the terms over the documents. The matrix product XX^T contains all these dot products. Element (i,p) (which is equal to element (p,i)) contains the dot product $\mathbf{t}_i^T \mathbf{t}_p$ ($= \mathbf{t}_p^T \mathbf{t}_i$). Likewise, the matrix $X^T X$ contains the dot products between all the document vectors, giving their correlation over the terms: $\mathbf{d}_j^T \mathbf{d}_q = \mathbf{d}_q^T \mathbf{d}_j$.

Now assume that there exists a decomposition of X such that U and V are orthogonal matrices and Σ is a diagonal matrix. This is called a singular value decomposition (SVD):

$$X = U \Sigma V^T$$

The matrix products giving us the term and document correlations then become

$$\begin{aligned} XX^T &= (U\Sigma V^T)(U\Sigma V^T)^T = (U\Sigma V^T)(V^{T^T}\Sigma^T U^T) = U\Sigma V^T V \Sigma^T U^T = U\Sigma\Sigma^T U^T \\ X^T X &= (U\Sigma V^T)^T(U\Sigma V^T) = (V^{T^T}\Sigma^T U^T)(U\Sigma V^T) = V\Sigma^T U^T U \Sigma V^T = V\Sigma^T\Sigma V^T \end{aligned}$$

Since $\Sigma\Sigma^T$ and $\Sigma^T\Sigma$ are diagonal we see that U must contain the eigenvectors of XX^T, while V must be the eigenvectors of $X^T X$. Both products have the same non-zero eigenvalues, given by the non-zero entries of $\Sigma\Sigma^T$, or equally, by the non-zero entries of $\Sigma^T\Sigma$. Now the decomposition looks like this:

$$\begin{array}{ccccccccc} & & X & & & U & & \Sigma & & V^T \\ & & (\mathbf{d}_j) & & & & & & & (\hat{\mathbf{d}}_j) \\ & & \downarrow & & & & & & & \downarrow \\ (\mathbf{t}_i^T) \rightarrow & \begin{bmatrix} x_{1,1} & \dots & x_{1,n} \\ \vdots & \ddots & \vdots \\ x_{m,1} & \dots & x_{m,n} \end{bmatrix} & = & (\hat{\mathbf{t}}_i^T) \rightarrow & \begin{bmatrix} \begin{bmatrix} \\ \mathbf{u}_1 \\ \\ \end{bmatrix} & \dots & \begin{bmatrix} \\ \mathbf{u}_l \\ \\ \end{bmatrix} \end{bmatrix} & \cdot & \begin{bmatrix} \sigma_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_l \end{bmatrix} & \cdot & \begin{bmatrix} [& &] \\ & \mathbf{v}_1 & \\ [& &] \end{bmatrix} \\ & & & & & & & & & \end{array}$$

The values $\sigma_1, \dots, \sigma_l$ are called the singular values, and $u_1, \dots, u_l$ and $v_1, \dots, v_l$ the left and right singular vectors. Notice the only part of U that contributes to $\mathbf{t}_i$ is the i'th row. Let this row vector be called $\hat{\mathrm{t}}_i$. Likewise, the only part of V^T that contributes to $\mathbf{d}_j$ is the j'th column, $\hat{\mathrm{d}}_j$. These are *not* the eigenvectors, but *depend* on *all* the eigenvectors.

It turns out that when you select the k largest singular values, and their corresponding singular vectors from U and V, you get the rank k approximation to X with the smallest error (Frobenius norm). This approximation has a minimal error. But more importantly we can now treat the term and document vectors as a "semantic space". The vector $\hat{\mathbf{t}}_i$ then has k entries mapping it to a lower dimensional space dimensions. These new dimensions do not relate to any comprehensible concepts. They are a lower dimensional approximation of the higher dimensional space. Likewise, the vector $\hat{\mathbf{d}}_j$ is an approximation in this lower dimensional space. We write this approximation as

$$X_k = U_k \Sigma_k V_k^T$$

You can now do the following:

- See how related documents j and q are in the low dimensional space by comparing the vectors $\Sigma_k \hat{\mathbf{d}}_j$ and $\Sigma_k \hat{\mathbf{d}}_q$ (typically by cosine similarity).
- Comparing terms i and p by comparing the vectors $\Sigma_k \hat{\mathbf{t}}_i^T$ and $\Sigma_k \hat{\mathbf{t}}_p^T$.
- Documents and term vector representations can be clustered using traditional clustering algorithms like k-means using similarity measures like cosine.
- Given a query, view this as a mini document, and compare it to your documents in the low dimensional space.

To do the latter, you must first translate your query into the low dimensional space. It is then intuitive that you must use the same transformation that you use on your documents:

$$\hat{\mathbf{d}}_j = \Sigma_k^{-1} U_k^T \mathbf{d}_j$$

Note here that the inverse of the diagonal matrix Σ_k may be found by inverting each nonzero value within the matrix.

This means that if you have a query vector q, you must do the translation $\hat{\mathbf{q}} = \Sigma_k^{-1} U_k^T \mathbf{q}$ before you compare it with the document vectors in the low dimensional space. You can do the same for pseudo term vectors:

$$\mathbf{t}_i^T = \hat{\mathbf{t}}_i^T \Sigma_k V_k^T$$
$$\hat{\mathbf{t}}_i^T = \mathbf{t}_i^T V_k^{-T} \Sigma_k^{-1} = \mathbf{t}_i^T V_k \Sigma_k^{-1}$$
$$\hat{\mathbf{t}}_i = \Sigma_k^{-1} V_k^T \mathbf{t}_i$$

Applications

The new low dimensional space typically can be used to:

- Compare the documents in the low dimensional space (data clustering, document classification).
- Find similar documents across languages, after analyzing a base set of translated documents (cross language retrieval).
- Find relations between terms (synonymy and polysemy).
- Given a query of terms, translate it into the low dimensional space, and find matching documents (information retrieval).
- Find the best similarity between small groups of terms, in a semantic way (i.e. in a context of a knowledge corpus), as for example in multi choice questions MCQ answering model.

Synonymy and polysemy are fundamental problems in natural language processing:

- Synonymy is the phenomenon where different words describe the same idea. Thus, a query in a search engine may fail to retrieve a relevant document that does not contain the words which appeared in the query. For example, a search for "doctors" may not return a document containing the word "physicians", even though the words have the same meaning.
- Polysemy is the phenomenon where the same word has multiple meanings. So a search may retrieve irrelevant documents containing the desired words in the wrong meaning. For example, a botanist and a computer scientist looking for the word "tree" probably desire different sets of documents.

Commercial applications

LSA has been used to assist in performing prior art searches for patents.

Applications in human memory

The use of Latent Semantic Analysis has been prevalent in the study of human memory, especially in areas of free recall and memory search. There is a positive correlation between the semantic similarity of two words (as measured by LSA) and the probability that the words would be recalled one after another in free recall tasks using study lists of random common nouns. They also noted that in these situations, the inter-response time between the similar words was much quicker than between dissimilar words. These findings are referred to as the Semantic Proximity Effect.

When participants made mistakes in recalling studied items, these mistakes tended to be items that were more semantically related to the desired item and found in a previously studied list. These prior-list intrusions, as they have come to be called, seem to compete with items on the current list for recall.

Another model, termed Word Association Spaces (WAS) is also used in memory studies by collecting free association data from a series of experiments and which includes measures of word relatedness for over 72,000 distinct word pairs.

Implementation

The SVD is typically computed using large matrix methods (for example, Lanczos methods) but may also be computed incrementally and with greatly reduced resources via a neural network-like approach, which does not require the large, full-rank matrix to be held in memory. A fast, incremental, low-memory, large-matrix SVD algorithm has recently been developed. MATLAB [4] and Python [5] implementations of these fast algorithms are available. Unlike Gorrell and Webb's (2005) stochastic approximation, Brand's algorithm (2003) provides an exact solution.

Limitations

Some of LSA's drawbacks include:

- The resulting dimensions might be difficult to interpret. For instance, in

  {(car), (truck), (flower)} --> {(1.3452 * car + 0.2828 * truck), (flower)}

  the (1.3452 * car + 0.2828 * truck) component could be interpreted as "vehicle". However, it is very likely that cases close to

  {(car), (bottle), (flower)} --> {(1.3452 * car + 0.2828 * **bottle**), (flower)}

  will occur. This leads to results which can be justified on the mathematical level, but have no interpretable meaning in natural language.
- LSA cannot capture polysemy (i.e., multiple meanings of a word)[*citation needed*]. Each occurrence of a word is treated as having the same meaning due to the word being represented as a single point in space. For example, the occurrence of "chair" in a document containing "The Chair of the Board" and in a separate document containing "the chair maker" are considered the same. The behavior results in the vector representation being an *average* of all the word's different meanings in the corpus, which can make it difficult for comparison. However, the effect is often lessened due to words having a predominant sense throughout a corpus (i.e. not all meanings are equally likely).
- Limitations of bag of words model (BOW), where a text is represented as an unordered collection of words.
- The probabilistic model of LSA does not match observed data: LSA assumes that words and documents form a joint Gaussian model (ergodic hypothesis), while a Poisson distribution has been observed. Thus, a newer alternative is probabilistic latent semantic analysis, based on a multinomial model, which is reported to give better results than standard LSA.

References

[1] http://en.wikipedia.org/w/index.php?title=Template:Semantics&action=edit
[2] http://patft.uspto.gov/netacgi/nph-Parser?patentnumber=4839853
[3] Markovsky I. (2012) Low-Rank Approximation: Algorithms, Implementation, Applications, Springer, 2012, ISBN 978-1-4471-2226-5
[4] http://web.mit.edu/~wingated/www/resources.html
[5] http://radimrehurek.com/gensim

- Thomas Landauer, Peter W. Foltz, & Darrell Laham (1998). "Introduction to Latent Semantic Analysis" (http://lsa.colorado.edu/papers/dp1.LSAintro.pdf) (PDF). *Discourse Processes* **25** (2–3): 259–284. doi: 10.1080/01638539809545028 (http://dx.doi.org/10.1080/01638539809545028).
- Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, Richard Harshman (1990). "Indexing by Latent Semantic Analysis" (http://lsi.research.telcordia.com/lsi/papers/JASIS90.pdf) (PDF).

Journal of the American Society for Information Science **41** (6): 391–407. doi: 10.1002/(SICI)1097-4571(199009)41:6<391::AID-ASI1>3.0.CO;2-9 (http://dx.doi.org/10.1002/(SICI)1097-4571(199009)41:6<391::AID-ASI1>3.0.CO;2-9). Original article where the model was first exposed.

- Michael Berry, Susan T. Dumais, Gavin W. O'Brien (1995). *Using Linear Algebra for Intelligent Information Retrieval* (http://citeseer.ist.psu.edu/berry95using.html). (PDF) (http://lsirwww.epfl.ch/courses/dis/2003ws/papers/ut-cs-94-270.pdf). Illustration of the application of LSA to document retrieval.
- "Latent Semantic Analysis" (http://iv.slis.indiana.edu/sw/lsa.html). InfoVis.
- Fridolin Wild (November 23, 2005). "An Open Source LSA Package for R" (http://cran.at.r-project.org/web/packages/lsa/index.html). CRAN. Retrieved 2006-11-20.
- Thomas Landauer, Susan T. Dumais. "A Solution to Plato's Problem: The Latent Semantic Analysis Theory of Acquisition, Induction, and Representation of Knowledge" (http://www.welchco.com/02/14/01/60/96/02/2901.HTM). Retrieved 2007-07-02.

External links

Articles on LSA

- Latent Semantic Analysis (http://www.scholarpedia.org/article/Latent_semantic_analysis), a scholarpedia article on LSA written by Tom Landauer, one of the creators of LSA.

Talks and Demonstrations

- LSA Overview (http://videolectures.net/slsfs05_hofmann_lsvm/), talk by Prof. Thomas Hofmann (http://www.cs.brown.edu/~th/) describing LSA, its applications in Information Retrieval, and its connections to probabilistic latent semantic analysis.
- Complete LSA sample code in C# for Windows (http://www.semanticsearchart.com/researchLSA.html). The demo code includes enumeration of text files, filtering stop words, stemming, making a document-term matrix and SVD.

Implementations

Due to its cross-domain applications in Information Retrieval, Natural Language Processing (NLP), Cognitive Science and Computational Linguistics, LSA has been implemented to support many different kinds of applications.

- Sense Clusters (http://www.d.umn.edu/~tpederse/senseclusters.html), an Information Retrieval-oriented perl implementation of LSA
- S-Space Package (http://code.google.com/p/airhead-research/), a Computational Linguistics and Cognitive Science-oriented Java implementation of LSA
- Semantic Vectors (http://code.google.com/p/semanticvectors/) applies Random Projection, LSA, and Reflective Random Indexing to Lucene term-document matrices
- Infomap Project (http://infomap-nlp.sourceforge.net/), an NLP-oriented C implementation of LSA (superseded by semanticvectors project)
- Text to Matrix Generator (http://scgroup20.ceid.upatras.gr:8000/tmg/index.php/Main_Page), A MATLAB Toolbox for generating term-document matrices from text collections, with support for LSA
- Gensim contains a fast, online Python implementation of LSA for matrices larger than RAM.

Article Sources and Contributors

Latent semantic analysis *Source*: http://en.wikipedia.org/w/index.php?oldid=583387980 *Contributors*: Abductive, Allenchue, Andy Dingley, Atomota, AxelBoldt, Clearsem, Copito42, Ealdent, Facopad, FearedInLasVegas, Forderud, Gabr, Galwhaa, Geva, Gregman2, Gzabers, Hiding, Hike395, Hu12, Imarkovs, Imersion, Itsonlychand, JackZhou, Jakarr, Jamelan, JasonAQuest, Jasonyo, Jitse Niesen, Jjwiseman, John of Reading, Johnchallis, JonDePlume, Jonsafari, Joshmyer, Juggernaut the, KYPark, Kcarnold, Kku, Klangenfurt, LatentDrK, Lucas Boullosa, Martinl, Melcombe, Michael Hardy, Mjordan, NeuronExMachina, Nicolasbock, Night Gyr, Nils Grimsmo, Nowa, Pdturney, Physicistjedi, Project mosaic, Purplefeltangel, Qiq, Qwertyus, R'n'B, Rahulkj, Rama, Richataxon, Riclas, Rjwilmsi, Ronz, Ruchirasharma, Ruud Koot, RyanLeiTaiwan, Sam Hocevar, Sander123, Schtickwriter, Scottybowl, Sergey pankratiev, SimonFunk, Siri Keeton, Siruguri, Sminc, Stephan Leeds, Stevertigo, Sylenius, The wub, TripleF, Veinor, Vplagnol, Waitak, Wanderingstan, Watchsmart, Wrockca, Yahya Abdal-Aziz, 138 anonymous edits

License