Semantic Similarity from Corpora - Latent Semantic Analysis

Carlo Strapparava
FBK-Irst Istituto per la ricerca scientifica e tecnologica
I-38050 Povo, Trento, ITALY
strappa@fbk.eu

Overview

- *Latent Semantic Analysis (LSA) is a theory and a method for extracting and representing the contextual-usage meaning of words*
- It works through statistical computations on large text corpora.
- LSA is mainly used to compute similarity between text units *at any level*:
  - Whole documents
  - Any sub-document entities (sentences, phrases, n-grams, …)
  - Single words

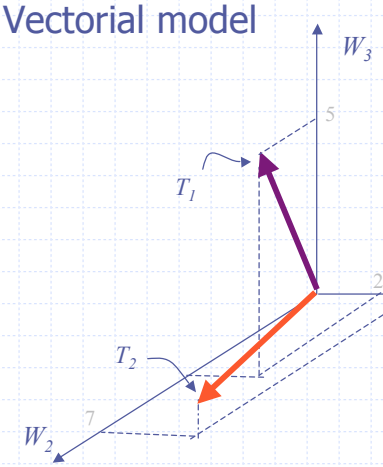
Documents and Words Space

- Let be $T = \{t_1, t_2, \ldots, t_n\}$ a corpus of documents, and $V = \{w_1, w_2, \ldots, w_k\}$ its vocabulary
- The vector space model VSM(T,V) is a k-dimensional space $\mathbf{R}^k$
- The text $t_j \in T$ is represented as a vector

$$\vec{t}_j = (x_1, \ldots, x_k) \text{ where } \begin{cases} x_z = freq(w_z, t_j) & \text{if } w_z \in t_j \\ x_z = 0 & \text{if } w_z \notin t_j \end{cases}$$

Vector Space Model

Vectorial model



$$\begin{pmatrix} & T_1 & T_2 & \ldots & T_n \\ W_1 & w_{11} & w_{12} & \ldots & w_{1n} \\ W_2 & w_{21} & w_{22} & \ldots & w_{2n} \\ \vdots & \vdots & \vdots & & \vdots \\ W_k & w_{k1} & w_{k2} & \ldots & w_{kn} \end{pmatrix}$$

Vector similarity

- Cosine measure between vectors

$$\cos(\vec{x},\vec{y}) = \frac{\vec{x}\cdot\vec{y}}{|\vec{x}||\vec{y}|} = \frac{\sum_i x_i y_i}{\sqrt{\sum_i x_i^2}\sqrt{\sum_i y_i^2}}$$

- For normalized vectors, the cosine is simply the dot product

$$\cos(\vec{x},\vec{y}) = \vec{x}\cdot\vec{y}$$

- Normalization is generally seen as a good thing: otherwise longer vectors would have an unfair advantage

Sample document collection

- A sample corpus *C* of Technical Report titles is drawn from two different sources (Human Computer Interaction and Graph Theory)
- It is possible to take into account also the term frequency
- For example: relevant title terms are selected for indexing if and only if appearing with *freq ≥ 2*

Sample document collection

Document id	*Document*
c1	Human machine interface for Lab ABC computer applications
c2	A survey of user opinion of computer system response time
c3	The EPS user interface management system
c4	System and human system engineering testing of EPS
c5	Relation of user-perceived response time to error measurement
m1	The generation of random, binary, unordered trees
m2	The intersection graph of paths in trees
m3	Graph minors IV: Widths of trees and well-quasi-ordering
m4	Graph minors: A survey

Sample document collection

Document id	*Document*
c1	Human machine interface for Lab ABC computer applications
c2	A survey of user opinion of computer system response time
c3	The EPS user interface management system
c4	System and human system engineering testing of EPS
c5	Relation of user-perceived response time to error measurement
m1	The generation of random, binary, unordered trees
m2	The intersection graph of paths in trees
m3	Graph minors IV: Widths of trees and well-quasi-ordering
m4	Graph minors: A survey

Terms by Documents matrix

	c1	c2	c3	c4	c5	m1	m2	m3	m4
human	1	0	0	1	0	0	0	0	0
interface	1	0	1	0	0	0	0	0	0
computer	1	1	0	0	0	0	0	0	0
user	0	1	1	0	1	0	0	0	0
system	0	1	1	2	0	0	0	0	0
response	0	1	0	0	1	0	0	0	0
time	0	1	0	0	1	0	0	0	0
EPS	0	0	1	1	0	0	0	0	0
survey	0	1	0	0	0	0	0	0	1
trees	0	0	0	0	0	1	1	1	0
graph	0	0	0	0	0	0	1	1	1
minors	0	0	0	0	0	0	0	1	1

Terms by Documents matrix

	c1	c2	c3	c4	c5	m1	m2	m3	m4
human	1	0	0	1	0	0	0	0	0
interface	1	0	1	0	0	0	0	0	0
computer	1	1	0	0	0	0	0	0	0
user	0	1	1	0	1	0	0	0	0
system	0	1	1	2	0	0	0	0	0
response	0	1	0	0	1	0	0	0	0
time	0	1	0	0	1	0	0	0	0
EPS	0	0	1	1	0	0	0	0	0
survey	0	1	0	0	0	0	0	0	1
trees	0	0	0	0	0	1	**1**	**1**	0
graph	0	0	0	0	0	0	**1**	**1**	1
minors	0	0	0	0	0	0	0	1	1

- Close meaning terms pairs, such as for ex. trees and graph appear in some documents

Terms by Documents matrix

	c1	c2	c3	c4	c5	m1	m2	m3	m4
human	**1**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**
interface	1	0	1	0	0	0	0	0	0
computer	1	1	0	0	0	0	0	0	0
user	**0**	**1**	**1**	**0**	**1**	**0**	**0**	**0**	**0**
system	0	1	1	2	0	0	0	0	0
response	0	1	0	0	1	0	0	0	0
time	0	1	0	0	1	0	0	0	0
EPS	0	0	1	1	0	0	0	0	0
survey	0	1	0	0	0	0	0	0	1
trees	0	0	0	0	0	1	1	1	0
graph	0	0	0	0	0	0	1	1	1
minors	0	0	0	0	0	0	0	1	1

- Other terms pairs, such as for ex. human and user, while semantically related, are not supported by the corpus

Singular Value Decomposition

- Let *A* be a matrix *m x n*, then there exists a factorization of the form

$$A_{m\times n} = U_{m\times r}\Sigma_{r\times r}V^T_{r\times n}$$

where

$$\Sigma = diag(\sigma_1,\ldots,\sigma_r)$$

$$\sigma_1 \geq \sigma_1 \geq \ldots \geq \sigma_r \geq 0 \text{ and } r = \min\{m,n\}$$

Reduced SVD

- In applications it is quite unusual to consider the full SVD
- Instead, it is often sufficient (as well as faster, and more economical for storage) to compute a reduced version of the SVD
- k-truncated SVD
  - Only the *k* column vectors of *U* and *k* row vectors of V^T corresponding to the *k* largest singular values Σ_r are calculated. The rest of the matrix is discarded
  - Of course the truncated SVD is no longer an exact decomposition of the original matrix *A*, but the approximate matrix A_k is in a very useful sense the closest approximation to *A* that can be achieved by a matrix of rank *k*.

SVD notation

- SVD computes new dimensions for the initial vector space
- *Truncated*-SVD selects only most significant dimensions describing both documents and terms
- Vectors in U and V represent respectively terms and documents as they are written in the new basis after SVD transformation

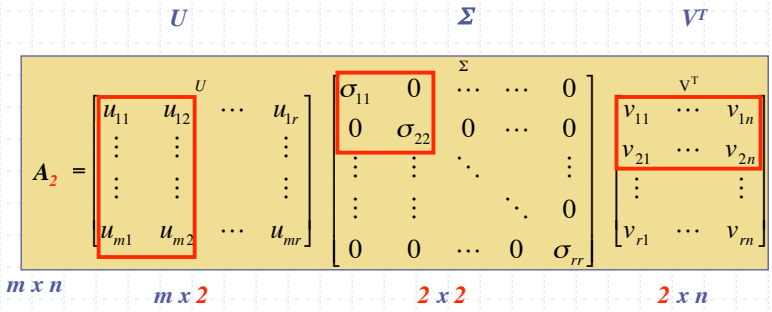
$$A = \underset{m \times r}{\underbrace{\begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1r} \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ u_{m1} & u_{m2} & \cdots & u_{mr} \end{bmatrix}}} \underset{r \times r}{\underbrace{\begin{bmatrix} \sigma_{11} & 0 & \cdots & \cdots & 0 \\ 0 & \sigma_{22} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & & \vdots \\ \vdots & \vdots & & \ddots & 0 \\ 0 & 0 & \cdots & 0 & \sigma_{rr} \end{bmatrix}}} \underset{r \times n}{\underbrace{\begin{bmatrix} v_{11} & \cdots & v_{1n} \\ v_{21} & \cdots & v_{2n} \\ \vdots & & \vdots \\ v_{r1} & \cdots & v_{rn} \end{bmatrix}}}$$

U Σ V^T

$m \times n$

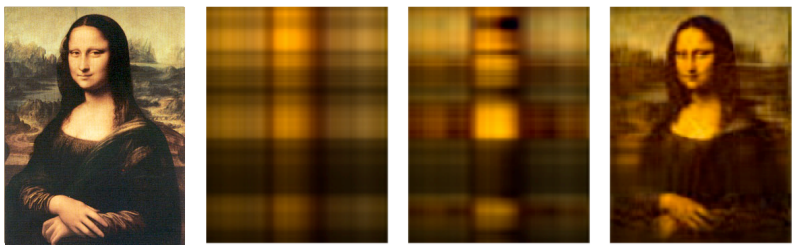
Truncated SVD

- SVD computes new dimensions for the initial vector space
- *Truncated*-SVD selects only most significant dimensions describing both documents and terms
- Vectors in U and V represent respectively terms and documents as they are written in the new basis after SVD transformation
- A_2 is an approximation of the original matrix A



SVD on an image

- The original matrix can then be reconstructed by adopting a fewer number of principal components

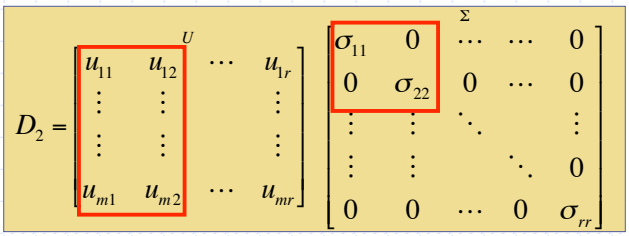


The original *1 singular value* *2 singular value* *16 singular value*

- A small number of dimensions are required to represent the original information, allowing a good quality reconstruction of the original picture

Truncated term vectors

- If only term vectors are considered you can use a more compact version (in practice some scaling through *idf* can be applied)



Intermezzo: *tf-idf*

- The tf-idf weight (term frequency-inverse document frequency) is a weight often used in information retrieval
- It is a measure used to evaluate how important a word is to a document.
- The importance increases proportionally to the *number of times a word appears in the document*, but is offset by *how common the word* is in all of the documents in the collection or corpus.
- tf-idf is often used by search engines to find the most relevant documents to a user's query.

Intermezzo: *tf-idf*

- Term frequency (tf) of a term t_i in a document d_j:

$$tf(t_i, d_j) = \frac{n_{t_i}}{\sum_{t \in d_j} n_t}$$

- Inverse document frequency is a measure of the *general importance* of the term (i.e. it is the logarithm of the number of all documents divided by the number of documents containing the term)

$$idf(t_i) = \log \frac{|D|}{|t_i \in d_j|}$$

- tf x idf: a high weight in tf-idf is reached by a high term frequency (in the given document) and a low document frequency of the term in the whole collection of documents: *the weights hence tends to filter out common terms*.

LSA: Singular Value Decomposition

- *Dimensionality reduction*: correlated terms and documents are collapsed into a common dimension in the LSA space

$A = U_{mxr} \Sigma_{rxr} V^T_{rxn}$

A =

	d1	d2	d3	d4	d5	d6
cosmonaut	1	0	1	0	0	0
astronaut	0	1	0	0	0	0
moon	1	1	0	0	0	0
car	1	0	0	1	1	0
truck	0	0	0	1	0	1

= mxr x rxr x rxn

U =

	dim1	dim2	dim3	dim4	dim5
cosmonaut	-0.44	-0.30	0.57	0.58	0.25
astronaut	-0.13	-0.33	-0.59	0.00	-0.61
moon	-0.48	-0.51	-0.37	0.00	-0.61
car	-0.70	0.35	0.15	-0.58	0.16
truck	-0.26	0.65	-0.41	0.58	-0.09

Σ =

2.16	0	0	0	0
0	1.59	0	0	0
0	0	1.28	0	0
0	0	0	1.00	0
0	0	0	0	0.39

Behind the numbers

- LSA represents:
  - the meaning of a word as a kind of average of the meaning of all the texts in which it appears
  - the meaning of a text as a kind of average of the meaning of all the words it contains
- The truncated SVD:
  - captures most of the important underlying structure in the association of terms and documents
  - removes the noise or variability in word usage that plagues word-based retrieval methods.

Applications - Information Retrieval

- Usual VSM-based IR brings documents exploiting just lexical match
- Latent Semantic Indexing allows IR based on semantic match
- A user query is transformed into a k-vector in the new space and compared to documents
- Now, all of Queries, Terms, and Documents "live" exactly in the same space, and LSA allows comparisons among any two of them
- Example: Information Filtering

Multi- Cross-language IR

- LSA is applicable to *any* language under very poor resource requirements (usually, just a text tokenizer is needed)
- When a corpus of multilingual parallel texts is available, LSA can be applied in a cross-language system.
  - A common multi-language Vector Space Model is built
  - A multilingual term-document matrix is computed and the k-truncated SVD is calculated
- User queries can be multilingual, allowing document matching in any language
- *We will see how we can relax the requirement to have a parallel corpus*

Term similarity

- LSA can act like unsupervised term clustering algorithms
- Possible applications:
  - Online Thesauri
  - Automatic extraction of index terms for document publication
  - …
- Term Similarity can be exploited for Information Retrieval again, implementing query-expansion techniques based on close-meaning terms found by LSA

LSA - some examples

- Ex. Find the similar words to "PC", "mother" and "mum"
- **pc**
  (("graphics#n" . 0.8055714) ("hardware#n" . 0.78060585)
  ("ram#n" . 0.7627977) ("floppy#a" . 0.75682265)
  ("peripheral#n" . 0.7536831) ("ms-dos#n" . 0.7410109)
  ("processor#n" . 0.7389588) ("macintosh#P" . 0.7323313)
  ("word_processor#n" . 0.71512634) ...)
- **mother**
  (("grandmother#n" . 0.6952353) ("mother#v" . 0.6228563)
  ("motherhood#n" . 0.6097218) ("aunt#n" 0.52301747)
  ("pregnant#a" . 0.5173134) ("childbirth#n" . 0.51611376)
  ("teat#n" . 0.5039671) ...)
- **mum**
  (("mummy#n" . 0.90080297) ("nanny#n" . 0.84046817)
  ("darling#n" . 0.8397952) ("granddad#n" . 0.83187544) ...)

User modelling

- Beyond document and term similarity, LSA has been successfully exploited for matching people on the basis of their respective interests.
- People are associated to documents they read and/or write by proper pseudo-document vectors
- *People-to-people* and *people-to-documents* comparisons are thus enabled, allowing
  - Aggregation of virtual communities sharing common interests
  - Finding proper experts on specified topics
  - Automatically assigning reviewers to conferences papers
  - …

Human Knowledge modeling

- LSA shows the capability of predicting the way in which humans deal with concepts
- Successful experiments have been carried out in several directions:
  - Prediction of text coherence and resulting comprehension
  - Prediction of subjective ratings of text properties, i.e. grades assigned to essays
  - Replication of semantic categorical clusterings of words found in certain neuropsychological deficits
  - Assessment of quality and quantity of knowledge contained in a text passage
- LSA has been used with good results to mimic synonymy, antonym, and other word relations

Human Knowledge modeling: the TOEFL experiment

- How well LSA captures synonymy compared to people?
- LSA's knowledge of synonyms was assessed with a standard vocabulary test
- 80-items questions of the standard ETS-TOEFL were used
- Given a single-word question, the test taker is asked to choose in a list the most similar word in meaning
- Ex. Given the problem word *levied* and four alternative words *imposed, believed, requested, correlated*
- LSA got 65% correct, identical to average score of most foreign students applying for colleges in USA

PMI-IR: Pointwise Mutual Information

- Turney (2001) introduces a simple *unsupervised learning algorithm* for evaluating the semantic similarity of the words
- PMI-IR: Pointwise Mutual Information to analyze statistical data collected by Information Retrieval
- The PMI-IR algorithm, like LSA, is based on co-occurrence.The core idea is that "a word is characterized by the company it keeps".
- There are many different measures of the degree to which two words co-occur. PMI-IR uses Pointwise Mutual Information

PMI-IR

- It is based on word co-occurrence using counts collected over very large corpora (e.g. the Web).
- Given two words w_1 and w_2, their PMI-IR is measured as:

$$PMI - IR(w_1, w_2) = \log_2 \frac{p(w_1 \,\&\, w_2)}{p(w_1) \times p(w_2)}$$

- Here, $p(w_1 \,\&\, w_2)$ is the probability that w_1 and w_2 co-occur.
- If w_1 and w_2 are statistically independent, then the probability that they co-occur is given by the product $p(w_1) \times p(w_2)$
- If they are not independent, and they have a tendency to co-occur, then $p(w_1 \,\&\, w_2)$ will be greater than $p(w_1) \times p(w_2)$
- Therefore PMI-IR is a measure of the degree of statistical dependence between w_1 and w_2

PMI-IR: calculation

- Turney (2001) suggests four types for calculating the score
- An interesting one is that using the *NEAR* query operator (co-occurrence within a ten-word window) provided for ex. by AltaVista

$$p_{NEAR} \cong \frac{hits(w_1\ NEAR\ w_2)}{WebSize} \qquad p(w_1) \cong \frac{hits(w_1)}{WebSize}$$

$$PMI - IR(w_1, w_2) = \frac{hits(w_1\ NEAR\ w_2) \times WebSize}{hits(w_1) \times hits(w_2)}$$

PMI-IR: evaluation

- Evaluated on the 80 TOEFL questions
- PMI-IR gets a score of 72.5% even better than LSA (that gets 65%)
- The average non-English college applicant usually gets 64.5%
- But when Landauer and Dumais made the experiments, they used an encyclopedia as text source not very large (matrix 61,000 x 30,473)

LSA pros and cons

- PROS:
  - Lower dimensional vectors
  - Good for ML algorithms that cannot handle high dimensional space
  - Each dimension is like a "latent" domain
- CONS:
  - New vectors are dense (we don't save memory)
  - High polysemous words confound LSA
  - Expensive to compute, but can be done offline

References

- Berry, Dumais, O'Brien. Using Linear Algebra for Intelligent Information Retrieval, 1994.
- Landauer, Foltz, Laham. An Introduction to Latent Semantic Analysis, Discourse Processes, 25, 259-284, 1998.
- Golub, Loan. Matrix Computations, Johns-Hopkins, Baltimore, second ed., 1989.
- Turney. Mining the Web for Synonyms. Lecture Notes in Computer Science 2163, 2001

Useful Packages

- SVDPACK comprises four numerical (iterative) methods for computing the singular value decomposition (SVD)
  - http://www.netlib.org/svdpack/
- SVDLIBC is a C library based on the SVDPACKC library. It offers a cleaned-up version of the code with a sane library interface
  - http://tedlab.mit.edu/~dr/SVDLIBC/