Latent Semantic Analysis

Laura Isla Navarro & Pascal Guldener

l.isla@campus.lmu.de guldener@cip.ifi.lmu.de

Referat

"Vertiefung der Grundlagen der Computerlinguistik"

Masterseminar WS2021

Thursday 4th February, 2021

Motivation

Semantic vs lexical search

Method

LSI

Example

Discussion

References

Prerequisites

- query: sequence of words/phrases
- corpus: collection of documents
- query -> corpus: *relevant* subset of documents

Lexical

- naive approach: raw term overlap
- uses lexical matching of query terms and terms in the document
- computes raw count similarity

Raw term overlap: Example

Query: Nasa lands rover on Mars

Corpus:

- **Nasa lands** rover and observes **Mars** (score: 3)
- Esa observes Venus with satellite (score: 0)
- **Mars lands** coup, buys Snickers (score: 2)

Raw term overlap: Problems

- Polysemy: terms with same appearance and different meanings
- Synonymy: terms with different appearance and same meaning
- => find a way to map meaning or conceptual topics into the same dimension in the vector space

Semantic

- in IR we want to retrieve docs which are relatively relevant to the query
- "relevant" as in semantically corresponding
- some form of meaning has to be captured at index time

Indexing conceptual topics

Latent Semantic Indexing (LSI):

- extract "latent" semantic structure (hidden by the ambiguous way we use language)
- use information revealed by co-occurence analysis of the terms
- represent these conceptual indexes in a vector space
- terms that are synonymous should be mapped to same dimension
- terms that are polysemous should be mapped to different dimensions (not possible on word level)
- reduce dimensionality in order to reduce noise

SVD

This is what Singular Value Decomposition does

- takes any rectangular shaped matrix A
- decomposes into a matrix of lower rank $\hat{A}$ such that

$$\Delta = |A - \hat{A}|_2 \tag{1}$$

where Δ is the minimal squared distance and $_2$ denotes the l2-norm

Decomposition with SVD

$A_{t \times d}$ is the correlation matrix of t terms and d documents

applying singular value analysis on A then yields decomposition

$$A_{t \times d} = T_{t \times n} S_{n \times n} D^T_{d \times n} \tag{2}$$

where

$n = min(t, d)$

S is the diagonal matrix of the singular values in descending order, all greater zero

T and D have orthonormal columns

Dimensionality reduction

small values in S contribute little we therefore reduce from the bottom and obtain a matrix of rank k

after reducing T and D accordingly we obtain:

$$\hat{A}_{t\times d} = T_{t\times k} S_{k\times k} D^T_{d\times k} \tag{3}$$

which is the smallest L_2-error approximation of A with respect to k

Choosing dimensionality

- S is a diagonal matrix of the singular values indicating the variation in co-ocurrence on each axis
- by reducing the k we prune away dimensions that contribute less to discrimination of topics
- but we might miss important topics
- or model too much noise in
- which leaves us with picking the right hyperparameter k

Choosing dimensionality

- Deerwester et al.[1] and subseqently Berry et al.[2] (on TREC datasets) reported best performance increase from 10 with maximum at 100 and decreasing afterwards
- Bradford (2008) [3] shows optimal performance at 400 for a 5 million doc collection, and stable results within 300 - 500 dimensions
- For most language simulations $500 < k < 1000$ optimal, with 300 being often best according to Landauer and Dumais [4]

Updating SVD decomposed data

maintaining a database of documents is a dynamic process we need to be able to add new docs

even for most sophisticated eigensolvers computing the decomposition is not feasible because polynomial
($\mathcal{O}(n^2k^3)$ where $n = min(t, d)$)[5]

given the size of contemporary databases

Updating SVD decomposed data

more efficent to "fold" new docs into latent space

when q is our query

$$\hat{q}_{k\times 1} = S^{-1}_{k\times k} T^{T}_{t\times k} q_{t\times 1} \tag{4}$$

$\hat{q}$ is then the sum of the multiplied termvectors with each dimension weighted

the projection $\hat{q}$ can then be used to find close documents in the latent space or to be appended to D as a column vector

Data

	d0	d1	d2	d3	d4	d5
bus	0	1	0	0	0	0
buys	0	0	0	0	0	1
circuit	1	0	0	0	0	0
cmos	0	1	0	0	0	0
coup	0	0	0	0	0	1
cpu	1	1	0	0	0	0
esa	0	0	0	0	1	0
lands	0	0	1	0	0	1
mars	0	0	1	1	0	1
nasa	1	0	1	0	0	0
observes	0	0	1	0	0	0
rover	0	0	1	1	0	0
satellite	0	0	0	1	1	0
snickers	0	0	0	0	0	1
venus	0	0	0	0	1	0

Term-Document incident matrix

Decomposition

we choose $k = 2$ since there are two topics

	0	1
bus	0.0	0.3
buys	0.2	-0.2
circuit	0.0	0.3
cmos	0.0	0.3
coup	0.2	-0.2
cpu	0.1	0.6
esa	0.0	-0.0
lands	0.4	-0.1
mars	0.6	-0.1
nasa	0.3	0.4
observes	0.2	0.1
rover	0.4	0.1
satellite	0.2	-0.0
snickers	0.2	-0.2
venus	0.0	-0.0

T_{15xk}

	0	1
0	2.8	0.0
1	0.0	2.0

S_{kxk}

	d0	d1	d2	d3	d4	d5
0	0.1	0.0	0.7	0.4	0.1	0.6
1	0.7	0.6	0.2	-0.0	-0.0	-0.4

D_{kx6}

Evaluation

Query: **Nasa lands rover on Mars**

- CPU circuit **nasa** (score: 0.31658441)
- CMOS bus cpu (score: 0.16366819)
- **Nasa lands rover** and observes **Mars** (score: 0.98994576)
- **mars rover** satellite (score: 0.97528171)
- Esa observes Venus with satellite (score: 0.84987205)
- **Mars lands** coup, buys Snickers (score: 0.76920703)

Computation

- real world corpora can have millions of documents, making full SVD not feasible
- LSA is performed on fraction of data and the rest is folded in
- this comes at the cost of losing accuracy when approximating the "true" model
- new words are ignored
- true co-occurrence patterns can only be measured when looking at the whole corpus

Advantages

- synonymy can be addressed by projecting terms that co-occur often close to each other
- high recall can be expected
- works well when overlap between query and documents is small
- can be used for topic clustering

Limitations

- compound terms treated as independent terms
- no obvious k
- can only be applied to corpora with "limited" polysemy
- high time complexity for SVD in dynamic collections
- precision can be worse than in more "naive" approaches

Conclusion

- whilst LSA is not a "silver bullet" when it comes to deal with the ambiguous ways we use language, it has proved to be useful in some cases by identifying underlying semantic structures.
- it relies on a robust mathematical framework to do so
- optimization criterion for reduction is clearly defined and efficiently computable

References I

[1] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *Journal of the American society for information science*, vol. 41, no. 6, pp. 391–407, 1990.

[2] M. W. Berry, S. T. Dumais, and G. W. O'Brien, "Using linear algebra for intelligent information retrieval," *SIAM review*, vol. 37, no. 4, pp. 573–595, 1995.

References II

[3] R. B. Bradford, “An empirical study of required dimensionality for large-scale latent semantic indexing applications,” in *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, ser. CIKM ’08, Napa Valley, California, USA: Association for Computing Machinery, 2008, pp. 153–162, ISBN: 9781595939913. DOI: 10.1145/1458082.1458105. [Online]. Available: https://doi.org/10.1145/1458082.1458105.

[4] T. K. Landauer and S. Dumais, “Latent semantic analysis,” *Scholarpedia*, vol. 3, no. 11, p. 4356, 2008, revision #142371. DOI: 10.4249/scholarpedia.4356.

References III

[5] X. Li, S. Wang, and Y. Cai, "Tutorial: Complexity analysis of singular value decomposition and its variants," *arXiv preprint arXiv:1906.12085*, 2019.