Nonnegative Tensor Factorization Usage to Find Semantic Distance Between Natural Language Texts

V.V. Smielov

The problem of finding the semantic distance between natural language texts is very important nowadays. As amount of words is huge and time performance is very important so the necessity of fast algorithm for preprocessing is very big. 3 demission array allows to transform natural language texts to mathematical object to work with. Nonnegative Tensor Factorization allows decrease size of matrix without loosing in speed performance.

Keywords: NTF, NLP, semantic distance.
ACM 2012: Computing methodologies → Artificial intelligence → Natural language processing → Lexical semantics.
MSC 2010: 15A23
UDC: 681.3

Introduction

Quite popular approach to construct algorithms for analyzing and processing natural language texts is to use vector and matrix models for linguistic data representation. Since texts can have huge size and words variety, the corresponding matrix representation of texts often reach extra large sizes. There is a need for economical and convenient representation of the received data sets. In the article the 3 dimensional array is proposed to act as main mathematical model, same as Nonnegative Tensor Factorization [1] as a suitable way to decrease size and an effective way of data transforming, and the mathematical model proposed and its software implementation.

Semantic Distance

Since the growth of information in text form is immense, more urgent problem arises automatic analysis of natural language texts. Today, there are many algorithms that allow for a primary word processing as defining the language of written text, define words that occur most often referring to a category of text according to given parameters. However, in analyzing texts in natural language is still a lot of unresolved issues. For some, there are algorithms that work on a particular set of inputs. Others have severe limitations on the amount of input. One of these problems is to determine the semantic distance between the set texts: the input program will receive two (or more) texts written in the same language; the exit program returns a factor indicating how objects, events or events described in texts semantically close to each other.

Semantic distance is a value that shows how the two concepts are connected (or similar) to each other. The calculation of semantic similarity is very widely used in computer linguistics, for example, semantic analysis, and anaphora resolution of

polysemy, clustering and classification of texts, identification of entities in text and so on. To determine the semantic proximity between texts requires an analysis of both texts, and on the basis of the analysis indicate the degree of closeness [2]. For the analysis of the text you want to switch from natural language to formal models. Such a model can be multi-dimensional matrix. A simpler approach is to use a two-dimensional term-document matrix. In such case main idea is to find "keywords" and frequency of their appearance in both texts [3], but this method does not allow make any difference between different senses of same word. So to establish more subtle and precise connections can use a three-dimensional matrix of subject – predicate – the application. But as the size of the matrix is very large scale, it raises the question of whether the methods tensor factorization integral oversized model transformation data presentation.

Mathematical Model of Block Nonnegative Tensor Factorization

The problem can be rewritten in such way:

Let $G \in \mathbf{R}^{R \times S \times T}$ be a third order nonnegative tensor to be analyzed. Nonnegative Tensor Factorization [4] of G requires solving a nonlinear minimization problem

$$min_{\hat{G} \geq 0} \| G - \hat{G} \|_F^2,$$

where $\hat{G}$ is the tensor of reconstructed data and $\| A \|_F^2$ is the square Frobenius norm. The rank-K reconstruction is defined by sums of tensor products:

$$\hat{G} = \sum_{k=1}^{K} u^{(k)} \otimes v^{(k)} \otimes w^{(k)},$$

where $u^{(k)} \in R^R$, $v^{(k)} \in R^S$ and $w \in R^T$ are basis vectors of nonnegative values. This reconstruction process is illustrated in Figure 1. The most commonly used

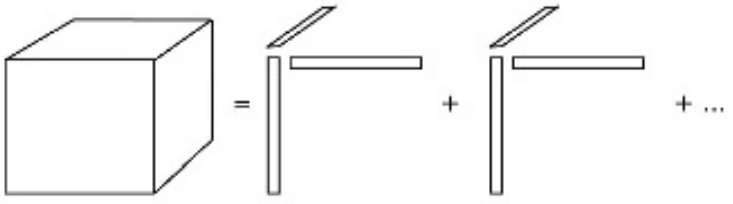

Figure 1. Principle of third order tensor factorization by using sums of rank-1 tensors

approaches to nonnegative tensor factorization are based on the Block Gauss-Seidel (BGS) method [5]. Using a combination of Gauss-Seidel and Jacobi iterative update schemes, these are calculated using iterative rules to update $u^{(k)}$, $v^{(k)}$ and $w^{(k)}$:

$$u_i^{(k)} \leftarrow \frac{u_i^k \sum_{s,t} G_{i,s,t} v_s^{(k)} w_t^{(k)}}{\sum_{m=1}^{K} u_i^{(k)} \langle v^{(m)}, v^{(k)} \rangle, \langle w^{(m)}, w^{(k)} \rangle}$$

$$v_i^{(k)} \leftarrow \frac{v_i^k \sum_{s,t} G_{r,i,t} u_s^{(k)} w_t^{(k)}}{\sum_{m=1}^{K} v_i^{(k)} \langle u^{(m)}, u^{(k)} \rangle, \langle w^{(m)}, w^{(k)} \rangle}$$

$$w_i^{(k)} \leftarrow \frac{w_i^k \sum_{s,t} G_{r,s,i} u_s^{(k)} v_t^{(k)}}{\sum_{m=1}^{K} w_i^{(k)} \langle u^{(m)}, u^{(k)} \rangle, \langle v^{(m)}, v^{(k)} \rangle}$$

where G is the data set and $\langle x; y \rangle$ denotes inner product. Usually, this iterative procedure must be repeated hundreds or even hundreds of thousands times to converge to the correct solution depending on the complexity of the data set. Therefore, iterative NTF computation is quite time consuming, and approaches to speeding it up would be useful.

An Algorithm for Block Nonnegative Tensor Factorization

Algorithm closely follows the theoretical description from previous section. The first step of the algorithm initializes the vectors u; v and w by using random values between 0 and 1. The NTF problem can be divided into such 3 subproblems, corresponding to written rules. Functions for their computation are named STEP. The inner products in the equations denominators can be calculated in advance and stored in $K \times K$ sized matrices. In Algorithm, these matrices are named M_u, M_v, and M_w, where $M_u = u^T u$, i.e.,

$$M_u = \begin{bmatrix} \langle u^{(1)}, u^{(1)} \rangle & \langle u^{(1)}, u^{(2)} \rangle & \dots & \langle u^{(1)}, u^{(K)} \rangle \\ \langle u^{(2)}, u^{(1)} \rangle & \langle u^{(2)}, u^{(2)} \rangle & \dots & \langle u^{(2)}, u^{(K)} \rangle \\ \vdots & \vdots & \ddots & \vdots \\ \langle u^{(K)}, u^{(1)} \rangle & \langle u^{(K)}, u^{(2)} \rangle & \dots & \langle u^{(K)}, u^{(K)} \rangle \end{bmatrix}$$

and M_v and M_w are defined similarly. The function for their computation is named CMAT in Algorithm. These matrices are symmetrical, so only the upper or lower triangle matrix needs to be calculated and stored.

Algorithm. Structure of the NTF

Require: G, K, I
1: init u, v, w
2: $M_u \leftarrow$ CMAT(u)
3: $M_v \leftarrow$ CMAT(v)
4: $M_w \leftarrow$ CMAT(w)
5: **for all** $i \in [0 \dots I-1]$ **do**
6: $u \leftarrow$ STEP (G, U, v, w, M_v, M_w)
7: $M_u \leftarrow$ CMAT(u)
8: $v \leftarrow$ STEP (G, U, v, w, M_u, M_w)

9: $M_v \leftarrow$ CMAT(v)
10: $w \leftarrow$ STEP(G, U, v, w, M_u, M_w)
11: $M_w \leftarrow$ CMAT(w)
12: **return** u, v, w

Calculating the numerator in the subproblem steps is the most time consuming operation. All other calculations, including creating the correlation matrices such as, do not take a significant amount of time in comparison. The numerator calculation consists mostly of repeated summing of a large array, so it is more demanding of memory bandwidth than it is computationally intensive. The subproblem steps only differ in the direction in which the layers of G are taken.

Integration of the Algorithm and Testing

As was said before, the main problem is to find the semantic distance between two texts. Therefor, we use stemming algorithm to process text and get stemmed words. (Stemming is the term used in linguistic morphology and information retrieval to describe the process for reducing inflected (or sometimes derived) words to their word stem, base or root form – generally a written word form. The stem need not be identical to the morphological root of the word; it is usually sufficient that related words map to the same stem, even if this stem is not in itself a valid root). After stemming we break text into sentences and find out diathesis [6] of each sentence. It can be whether active or passive voice. For active voice we take into consideration words that represent subject and action, for passive voice we take into consideration words that represent object and action.

Now we can build a matrix $G_{u,v,w}$ that will represent input data as the amount of times words v and w were appearing in same sentence in text u. Such model will allow easily respond to requests of type "find distance between text i and text j". But as amount of words can be huge we need decrease size of built matrix. That is why NTF was used. Such approach allowed decrease size of matrix without big difference in time to access an element of the matrix.

Finally, such approach allowed to consider as not semantically closed texts, that has been containing quite similar words as main but in different senses.

To provide tests on real data and prove points, set of pair of texts were selected. Experts (humans) defined for each pair whether texts are semantically close or not. Then on same pairs 2 different algorithms were applied. First method defined semantic distance according to distance between main words of texts. Second method defined semantic distance with usage 3 dimensional matrix to represent texts and approach described before. For both algorithm same border to define similarity was used (if semantic distance is less than defined border then we were assuming that texts are semantically closed and texts are not semantically closed otherwise). So there are 4 possible scenarios: experts have decided that texts are not close and program makes the same decision (True Negative), experts and program decided that texts are close (True Positive), experts decided that texts are close but result of program was positive (False Negative), and experts decided that texts are not semantically closed but program result was opposite (False Positive). Result of experiment

is provided in Table 1.

Table 1. Compare results of 2 methods

Result	main words	3 dimensional matrix
True Positive	89	88
True Negative	24	31
False Positive	17	9
False Negative	6	8

As we can see from results table amount of True Negative has increased and amount of False Positive has decreased, so generally algorithm become better.

Conclusions

Semantic distance between texts can be found as a distance between main words of the texts. In this article tree demission matrix (text-object-action) was proposed as a main mathematical model to represent a text. As the variety of words can be huge nonnegative tensor factorization was proposed to decrease size of matrix. Explanation of the main idea, mathematical proof and program realization idea were providing.

References

[1] A. Cichocki, R. Zdunek, A. Huy-Phan, and S. Amari, *Nonnegative Matrix and Tensor Factorizations Applications to Exploratory Multi-way Data Analysis and Blind Source Separation.* John Wiley and Sons, Ltd, 2009.

[2] S. Harispe, S. Ranwez, S. Janaqi, and S. Montmain, *Semantic Measures for the Comparison of Units of Language, Concepts or Entities from Text and Knowledge Base Analysis.* Arxiv Corr, 2013.

[3] J. Jiang, "Semantic similarity based on corpus statistics and lexical taxonomy," *In the Proceedings of ROCLING X*, 2007.

[4] J. Antikainen, J. Havel, R. Josth, A. Herout, P. Zemcek, and M. Hauta-Kasari, *Nonnegative Tensor Factorization Accelerated Using GPGPU.* CezmSMT,, 2011.

[5] L. Grippo and M. Sciandrone, "On the convergence of the blocknonlinear gauss-seidel method under convex constraints," *Operations Research Letters*, vol. 26, no. 3, pp. 127–136, 2003.

[6] W. O'Grady, J. Archibald, M. Aronoff, and J. Rees-Miller, *Contemporary Linguistics: An Introduction.* Bedord St. Martins, 2003.

Author

Valerii Viktorovych Smielov — the 1st year postgraduate student, Cybernetics Faculty, Taras Shevchenko National University of Kyiv, Kyiv, Ukraine; E-mail: *smielov.knu@gmail.com*