Applicability of Tensor Factorization methods for the problem of Predicate Induction

Madhav Nimishakavi
Indian Institute of Science
Bangalore, India
madhav@csa.iisc.ernet.in

Uday Singh Saini
Indian Institute of Science
Bangalore, India
uday.s.saini@gmail.com

Partha Talukdar
Indian Institute of Science
Bangalore, India
ppt@cds.iisc.ac.in

Abstract

Given a set of documents from a specific domain (e.g., medical research journals), how do we automatically identify the predicates, i.e., categories (e.g., *Patient*, *Surgery*) and relations (e.g., *undergoSurgery(Patient, Surgery)*), necessary to understand the given set of documents? We refer to this problem as *Predicate Induction (PI)*. Open Information Extraction (OIE) techniques aim at extracting surface-level text triples of the form *(John, had an, Angioplasty)*. While such OIE triples provide instances of predicates of interest, they don't induce the yet unknown predicates themselves. Tensors provide a natural representation for such triples, and factorization of such tensors provide a plausible solution for the PI problem. While many such factorization techniques exist in the literature (e.g., PARAFAC, Tucker, RESCAL, etc.), they haven't been applied to the problem of Predicate Induction from unstructured text. We fill this gap in this paper and present a comprehensive comparison of effectiveness of various tensor factorization methods for the PI problem. We report our findings on multiple real-world datasets. We hope to make all datasets and code publicly available upon publication of the paper.

1 Introduction

In order to understand the example text fragment shown below,

. . . John had angioplasty last Tuesday. . . .

a Natural Language Understanding (NLU) should be able to infer that this text snippet provides instantiations of two types of predicates: (1) categories: *Patient*, *Surgery*; and (2) relation: *undergoSurgey(Patient, Surgery)*. Once such predicates have been identified, recent advances in ontological Knowledge Graph (KG) construction methods (Dong et al., 2014; Mitchell et al., 2015) may be used to identify large number of instances of such predicates from Web-scale datasets. But the challenge is how one could identify such predicates and their signatures automatically from unstructured text without any supervision? We refer to this problem as *Predicate Induction (PI)*.

Open Information Extraction (OIE) techniques (Etzioni et al., 2011) aim to extract surface-level triples from unstructured text. For example, given the sentence shown above, an OIE system will extract the triple (*John, had, angioplasty*). Over the last few years, OIE has emerged as a preferred way to identify parts of the sentence containing factual knowledge, and store that in the form of a triple. However, given such triples, it is not clear how one might go about identifying the predicates, i.e., categories and relations among those categories generating such triples.

Tensors are a higher order generalization of matrices and they provide a natural way to represent the triple data generated by OIE extractions. Tensor factorization is an extensively studied area with several well known methods, e.g., PARAFAC, Tucker Decomposition, RESCAL (Kolda and Bader, 2009). Such techniques have in fact also been applied to factorize Knowledge Graphs whose ontology is already known (Nickel et al., 2012). Applying such tensor factorization methods over OIE triples to identify predicates is a natural approach, but one that has not been explored

so far. In this paper, we fill this gap and make the following contributions:

- We present a comprehensive comparison of multiple tensor factorization techniques for the Predicate Induction (PI) problem. To the best of our knowledge, this is the first such study of its kind.
- We perform our experiments on multiple real-world datasets and report the insights obtained in the process. Among others, we identify the need for non-negative factorization in order to induce higher-quality inductions.
- We finish the paper by identifying a set of open challenges in this important problem of predicate induction. We hope to make all the datasets and code publicly available upon publication of the paper.

Our goal in this paper is not to propose yet another new method, but to study the applicability of tensor factorization methods for the problem of Predicate Induction.

2 Related Work

A method for inducing (binary) relations and the categories they connect was proposed by (Mohamed et al., 2011). However, in that work, categories and their instances were known a-priori. In contrast, in case of PI, both categories and relations are to be induced.

Predicate Induction can be considered as a sub problem of Ontology Induction (Velardi et al., 2013), but instead of building a full fledged hierarchy we are particularly interested in finding signatures of predicates and shallow relations among them. Even though tensors provide a natural representation they have not been explored for these class of problems, so we try address those gaps in this paper.

A method for canonicalizing noun and relation phrases in OIE triples was recently proposed in (Galárraga et al., 2014). The main concentration of this approach is to cluster lexical variants of a *single* entity or relation. This is not directly relevant for PI, as they are interested in grouping entities of the same type into one cluster, and use that to induce relation schema.

(Chambers, 2013) propose a method for event schema induction, which is the task of learning high-level representations of complex events and their entity roles from unlabeled text. This approach is heavily dependent on Named Entity Recognition (NER) and hence may not be scalable for every domain. Our focus in this paper is on unsupervised tensor factorization methods.

Due to their flexibility of representation and effectiveness, tensor factorization methods have seen increased application in Knowledge Graph (KG) related problems over the last few years. Methods for decomposing ontological KGs such as YAGO (Suchanek et al., 2007) were proposed in (Nickel et al., 2012; Chang et al., 2014). In these cases, predicates are known in advance, while we are interested in inducing such predicates from unstructured text. Gigatensor, a scalable implementation of PARAFAC tensor factorization was proposed in (Kang et al., 2012).

3 Tensor Factorization Methods

In this section, we briefly review two popular tensor factorization methods and their non-negative variants which are compared in the experiments of this paper. All these methods take a tensor, $\mathcal{X}$, as an input which is a representation of the triples extracted from the corpus. We first describe how to construct the tensor from OIE triples, and describe various factorization methods that operate over this tensor.

3.1 Tensor Construction

Tensor is constructed from a tab delimited file of triples. For PARAFAC, an element x_{ijk} of the tensor $\mathcal{X}$ is an element corresponding to the triple formed by i^{th} subject, $j^{th}predicate$ and k^{th} object , score (scoring mechanism is explained in the next section) corresponding to this triple is stored in the tensor. If the triple with that combination is not present in the input file then zero is stored in tensor at that index.

Tensor construction for RESCAL is different from that of PARAFAC, here an element x_{ijk} refers to triplet formed by i^{th} noun phrase , j^{th} noun phrase and k^{th} verb phrase. In PARAFAC, subject noun phrases and object

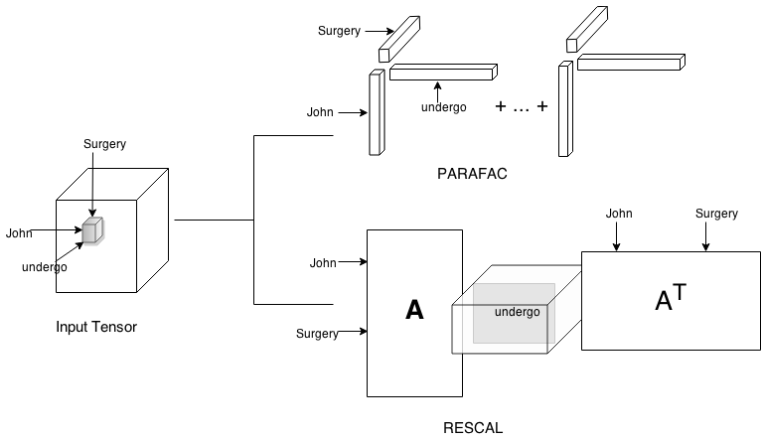

Figure 1: Decomposition of a Tensor by PARAFAC and RESCAL where *(John, undergo, Surgery)* is a triple.

noun phrases have separate indexing unlike for RESCAL in which all the noun phrases have same indexing. Figure 1 illustrates the decomposition of input tensor performed by PARAFAC and RESCAL.

3.2 Notations

Following are the notations used in this paper

Notation	Definition
$\mathcal{X}$	A Tensor
$X_{(n)}$	Mode-n matricization of a tensor
$\circ$	Outer product
$\odot$	Khatri-Rao product
$\otimes$	Kronecker product
$*$	Hadmard product
$A^\dagger$	Pseudoinverse of A

Table 1: Notations

3.3 PARAFAC

PARAFAC (or CANDECOMP) (Harshman, 1970) decomposes the tensor into a sum of component rank-one tensors. Let $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$, it will be decomposed to

$$\mathcal{X} \approx \sum_{r=1}^{R} a_r \circ b_r \circ c_r$$

where R is a positive integer and $a_r \in \mathbb{R}^I, b_r \in \mathbb{R}^J$ and $c_r \in \mathbb{R}^K$ for $r = 1, ..., R$. This can be written elementwise as

$$x_{ijk} = \sum_{r=1}^{R} a_i b_j c_k, \quad \forall 1 \leq i \leq I, \; 1 \leq j \leq J, 1 \leq k \leq K$$

Goal of PARAFAC is to compute a decomposition with R components that best approximate the given tensor, $\mathcal{X}$ i.e., to find

$$\min_{\hat{\mathcal{X}}} \| \mathcal{X} - \hat{\mathcal{X}} \| \text{ with}$$

$$\hat{\mathcal{X}} = \sum_{r=1}^{R} \lambda_r \times a_r \circ b_r \circ c_r = [\lambda; A, B, C]$$

Alternating Least Square (ALS) (Harshman, 1970) can be used to solve the factorization , which fixes B and C to solve A, then fixes A and C for B, then fixes A and B to solve for C and continues to repeat the entire procedure until some convergence criterion is satisfied. Update for A is as shown below:

$$\hat{A} = X_{(1)}(C \odot B)(C^T C * B^T B)^\dagger$$

Updates for B and C are defined similarly. We used MATLAB Tensor Toolbox Version 2.6 (Bader et al., 2015) for computing the factorization by ALS.

3.4 Non-Negative PARAFAC (NN-PARAFAC)

The objective of Non-Negative PARAFAC (NN-PARAFAC) is same as PARAFAC, with the additional non-negative constraints $A \geq 0, B \geq 0, C \geq 0$. Non-Negative PARAFAC uses the NMU algorithm (Lee and Seung, 2000) for updates. We used MATLAB Tensor Toolbox Version 2.6 (Bader et al., 2015) to compute the factorization using NMU updates.

3.5 RESCAL

RESCAL (Nickel et al., 2011) decomposes a tensor, $\mathcal{X} \in \mathbb{R}^{n \times n \times m}$, into a factor matrix $A \in \mathbb{R}^{n \times r}$ and a core tensor $\mathcal{R} \in \mathbb{R}^{r \times r \times m}$. Where r is the number of latent factors. Each slice $X_k, k = 1, ..., m$ of the $\mathcal{X}$ can be interpreted as an adjacency-matrix between n entities of the relation and each slice is factored into

$$X_k = AR_kA^T, \ \forall k = 1, ..., m$$

The factor matrices A and R_k can be computed by solving the regularized minimization problem

$$\min_{A,R_k} f(A, R_k) + g(A, R_k)$$

where

$$f(A, R_k) = \frac{1}{2}(\sum_k \parallel \mathcal{X}_k - AR_kA^T \parallel_F^2)$$

and g is the following regularized term

$$g(A, R_k) = \frac{1}{2}(\parallel A \parallel_F^2 + \sum_k \parallel R \parallel_F^2)$$

The updates for A and R_k are given by

$$A \leftarrow [\sum_{k=1}^{m} X_kAR_k^T + X_k^TAR_k][\sum_{k=1}^{m} B_k + C_k + \lambda I]^{-1}$$

where

$$B_k = R_kA^TAR_K^T, C_k = R_k^TA^TAR_K$$

and $R_k \leftarrow (Z^TZ + \lambda I)^{-1}Zvec(X_k)$

where $Z = A \otimes A$

3.6 Non-Negative RESCAL (NN-RESCAL)

Non-Negative Rescal (Krompaß et al., 2013)implements non-negative tensor decompositions based on RESCAL by employing multiplicative update rules for A and $\mathcal{R}$. Objective function of Non-Negative Rescal is same as that of RESCAL, which is

$$\min_{A,R_k}(\sum_k \parallel \mathcal{X}_k - AR_kA^T \parallel_F^2) + (\lambda_A \parallel A \parallel_F^2 + \lambda_R \sum_k \parallel R \parallel_F^2)$$

The updates are given by

$$A \leftarrow A \bullet \frac{\sum_k X_kAR_k^T + X_k^TAR_k}{A([\sum_k R_kA^TAR_k^T + R_k^TA^TAR_k] + \lambda_A I)}$$

$$R_k \leftarrow R_k \bullet \frac{A^TX_kA}{A^TAR_kA^TA + \lambda_R R_k}$$

Table 2: Datasets used in the experiments.

Dataset	# Documents	# Triples
Ohsumed	50,216	535,784
NYT Sports	20,940	499,068

4 Experiments

4.1 Experimental Setup

Datasets: We used two datasets for the experiments in this paper, they are summarized in Table 2.

- **Ohsumed**: The Ohsumed collection is a subset of the MEDLINE database, which is a bibliographic database of peer-reviewed medical literature maintained by the National Library of Medicine. The collection contains 50,216 medical abstracts.
- **NYT Sports**: The New York Times annotated corpus contains over 1.8 million articles written and published by the New York Times between 1987 and 2007. For our experiments we considered a collection of 20,940 documents related to Sports category published between 2005 and 2007.

Open IE Triple Extraction: We used Open IE v4.0[1] to extract triples from the datasets described above. Number of triples extracted from each dataset is shown in Table 2. Following processing steps were carried out during and after triple extraction:

- Stanford CoreNLP (Manning et al., 2014) was used for coreference resolution. This was performed over the raw corpus before OIE extraction to make sure there were no pronouns in triple arguments.
- Justeson and Katz filter (Justeson and Katz, 1995) was applied over triple arguments to extract base NPs from arguments with extraneous tokens.
- Duplicate triples were removed, and verb phrases in the triples were lemmatized.

[1] Open IE v4.0: `http://knowitall.github.io/openie/`

Table 3: FIT comparison between binary and triple-scored tensor decomposition by RESCAL in the Ohsumed triple tensor. Using triple scores results in better decomposition (higher fit). [see Section 4.2]

Dimensions (R)	Binary	Triple-scored
25	0.031	**0.053**
50	0.044	**0.069**
75	0.048	**0.082**
100	0.062	**0.084**
125	0.065	**0.092**

All the numbers in triple arguments were normalized with a keyword $< NUM >$.

Evaluation Metric: Given tensor $\mathcal{X}$, let $\hat{\mathcal{X}}$ be its reconstruction generated from the decomposition produced by one of the methods described in Section 3. We measure the quality of this reconstruction using the following fit function.

$$\text{FIT}(\mathcal{X}, \hat{\mathcal{X}}) = 1 - \frac{||\mathcal{X} - \hat{\mathcal{X}}||_F}{||\mathcal{X}||_F} \qquad (1)$$

where $||\mathcal{X}||_F = \sqrt{\sum_{i,j,k} x_{i,j,k}^2}$ is the Frobenius norm.

4.2 Binary vs Triple-scored Tensor

We note that the value of a triple represented by the indices (i, j, k) in the tensor $\mathcal{X}$ is given by the cell $x_{i,j,k}$. We consider two scoring schemes:

- **Binary**: In this case, the tensor cell value corresponding to any valid triple is 1, and 0 otherwise, i.e., $x_{i,j,k} \in \{0, 1\},\ \forall i, j, k$
- **Triple-scored**: All triples are not equally likely, and the same is true for the arguments in each triple. We consider the following triple scoring scheme:

$$\begin{aligned} x_{i,j,k} \quad = \quad & \Gamma - \log(\#(a1) \times \#(a2)) \\ & -\text{LEN}(a1) - \text{LEN}(a2) \end{aligned}$$

where $a1$ and $a2$ are the first and second arguments of the triple, $\#(a1)$ is frequency of $a1$ in the triple set, $\text{LEN}(a1)$ is the string length of argument $a1$, and Γ is a positive number added to make the score positive. This score is intended to discount triples with very frequent arguments, and also triples with argument segmentation errors which usually result in long argument string.

Table 4: FIT comparison between RESCAL & NN-RESCAL for Ohsumed data. Non-negativity results in reduced Fit across all latent ranks (R), but it improves latent factor interpretability. See Section 4.4 for details.

Rank (R)	NN-RESCAL	RESCAL	Fit Change (%)
25	0.049	**0.062**	22 %
50	0.068	**0.081**	17%
75	0.079	**0.093**	17 %
100	0.082	**0.103**	24 %
125	0.091	**0.111**	21 %

Experimental results comparing factorization of binary vs triple-scored tensors is presented in Table 3. Across all settings, we find that using triple scores results in better factorization (higher fit). Unless otherwise stated, we use triple scoring in all subsequent experiments.

4.3 Hyperparameter Sensitivity

We tested the tensor factorization methods over a wide range of hyperparameter values, but didn't find them to be sensitive to such variation. For example, in case of NN-RESCAL, the FIT changed only from 0.049 to 0.051 when the hyperparameter varied from 0.01 to 1 (two orders of magnitude change).

4.4 Result #1: Non-Negativity improves Interpretability

In this section, we are interested in evaluating the importance of non-negative constraints during tensor decomposition of OIE triples. For this, we compared the FIT scores of the RESCAL and NN-RESCAL methods on the Ohsumed triple tensor. Results are shown in Table 4. From this table, we observe that RESCAL results in a better decomposition across all latent ranks. We observe similar trends in the NYT dataset and while comparing PARAFAC with NN-PARAFAC. However, we find that inspite of the reduced fit

due to the non-negativity constraints, it ultimately results in more interpretable latent factors. Additionally, non-negativity helps us overcome the *overgeneration problem* which we describe next.

Overcoming overgeneration using non-negativity: To understand the problem of overgeneration and its relationship to interpretability, let us consider the example of a specific triple *(patients, receive, chemotherapy)* from the Ohsumed dataset. We first remind the reader that RESCAL decomposes the slice of tensor $\mathcal{X}$ corresponding to textual relation *receive* as

$$\mathcal{X}_{receive} \sim AR_{receive}A^T$$

From this decomposition, we find that $R_{receive}(22, 25)$ is much higher than any other cell in R, followed by $R_{receive}(22, 21)$. This means that the *receive* relation induced by this decomposition connects the 22^{nd} column of A (latent factor) with the 25^{th} column of A. Similarly, between 22^{nd} and 21^{st} columns of A.

Looking at the columns of A (latent factors), we find that *patients* is present with high positive score in columns 22 and 25. Since, $R_{receive}(22, 25)$ is active, because of this dual presence, the decomposition will end up generating the tuple *(patients, receive, patients)*, a case of *over-generation*, as this triple is not present in the input tensor $\mathcal{X}$. Thus to stop this invalid tuple from getting generated, RESCAL will have to compensate for this overproduction by generating the same tuple with a corresponding negative score. In fact, we found this exactly to be the case! We observed that *patients* is also active in latent factor 21, but with a high negative score. From above, we know that $R_{receive}(22, 21)$ is also active, and hence this generates the *(patients, receive, patients)* tuple but this time with a negative score to counter for the overgeneration. Thus, due to the relatively under-constrained nature, RESCAL ends up overgenerating and then compensating for it. Unfortunately, this dilutes the interpretability of the latent factors as any phrase may be added to a latent factor as long as this incorporation may be negated through some other factor.

This prompted us to look at the interpretability properties of RESCAL's non-negative counterpart, NN-RESCAL. Indeed, we find that NN-RESCAL doesn't suffer from the overgeneration problem as it doesn't have the option of having negative entries in latent factors, and thereby no overgeneration as well as no compensation for it. Non-negativity forces the model to commit to specific semantics of the latent factors, as it has lesser degrees of freedom. We indeed find that the latent factors of A in case of NN-RESCAL are significantly more interpretable than that of RESCAL. As we shall see in Section 4.6, this increased interpretability allows for more number of accurate predicates to be induced.

Similar benefits of non-negativity on interpretability have also been observed in matrix factorization (Murphy et al., 2012).

4.5 Result #2: PARAFAC can only induce small number of predicates

A very popular heuristic deployed to find an appropriate rank of a PARAFAC decomposition is a diagnostic measure called CorConDia (Bro and Kiers, 2003). Finding an appropriate number for rank of a PARAFAC decomposition is a fairly practical step to take if one is to exploit the uniqueness property of PARAFAC (Kolda and Bader, 2009). We used the CorConDia implementation given by (Papalexakis and Faloutsos, 2015). [2]

For varying rank, a high CorConDia score (say, 90+) implies that PARAFAC is a good model at that rank for the data. However, rank of the first big drop in CorConDia score indicates the maximum rank that may be modeled using PARAFAC, any higher rank modeling should be performed using Tucker models, such as RESCAL. CorConDia scores of PARAFAC decompositon of the Ohsumed dataset is shown in Table 5. From this we find that PARAFAC is only able to induce maximum 3 predicates. We find this to be a constant problem with PARAFAC across multiple datasets. This suggests exploration of Tucker models such as RESCAL for this problem. This analysis was one of the main motivations for us to explore RESCAL/Tucker after PARAFAC decomposition.

[2] http://www.cs.cmu.edu/~epapalex/src/efficient_corcondia.zip

Table 5: CorConDia scores of PARAFAC on Ohsumed decomposition. The first big drop in CorConDia score gives an estimate of the number of valid latent factors in the decomposition. This number is 4 in this case. In general, we find PARAFAC to be able to support on a small number of ranks in our data decompositions. See Section 4.5 for details.

Rank	1	2	3	4	5	6	7	8	9
CorConDia Score	100	100	99.9	-5.0	53.9	-275	-1447	-39541.7	-65235. 8

	Ohsumed			NYT		
Method	Predicted #rel instances	Avg. # correct instances	Accuracy	Predicted #rel instances	Avg. # correct instances	Accuracy
PARAFAC	25	10	40%	25	3.5	14%
NN-PARAFAC	25	15.33	**61%**	25	8	32%
RESCAL	20	2.67	13%	12	1	8.3%
NN-RESCAL	20	7.68	38.3%	10	5.5	**55%**

Table 6: Predicate induction accuracies (averaged over two human annotator judgments) produced by the four tensor factorization methods over two datasets. Accuracy of best method in each dataset is marked in bold. See Section 4.6 for details.

4.6 Result #3: NN-RESCAL is the most effective method for Predicate Induction

In general, we observe that an under-constrained model (such as PARAFAC or RESCAL) tends to achieve higher fit value compared to their non-negative counterparts, viz., NN-PARAFAC and NN-RESCAL. However, this increased fit comes at a cost of prediction interpretability as we shall see shortly. In this section, we are particularly interested in evaluating correctness of the predicate instances induced by various algorithms from the two datasets.

A human annotator was presented with the induced relation by one of the four algorithms, along with top ranking example words from each of the two latent factors. The annotator was asked to label the latent factors in the context of the relation. Failure to do so, was considered as prediction of an invalid predicate. Accuracy was then collected by averaging scores from annotators. Experimental results are shown in Table 6. From this table, even though there is no single algorithm that performs well in all settings, we observe that non-negativity constrained models tend to produce more coherent predicates as opposed to their under-constrained alternatives.

Examples of a few relations induced by NN-RESCAL from the NYT Sports and Ohsumed datasets are shown in Table 7. Please note that the argument labels in this example are provided by the human annotators. Comparing NN-PARAFAC with NN-RESCAL, we find that NN-RESCAL is able to induce much more diverse relation predicates, compared to NN-PARAFAC (and also PARAFAC) which produces relations with much reduced diversity (often just one or two different relations). Due to the quality and diversity of inductions, we consider NN-RESCAL to be the single best method among the four alternative considered for predicate induction.

4.7 Open Challenges

Challenge #1: Automatic labeling of the latent factors is a hard problem. We tried an approach similar to (Lau et al., 2011) using Wikipedia category information. The labeling scheme worked well for some of the latent factors such as [*surgery, appendectomy, laparotomy, splenectomy*] for which the label prediction was *surgery*, but for some of the latent factors like [*positive, negative, seropositive, seronegative*], which can be *result of a medical test*, labeling didn't work as well. Automatic labeling of latent factors with abstract concepts remains an open problem.

Challenge #2: While OIE triples provide a

NYT Sports Dataset				
Predicted Relation	Argument Labels	Top Words		
score	Player Point	Carter Num points	O'neal 2 Num points	Vince Carter Num points
left	Time Game	Num seconds game	Minutes half	Num minutes regulation
hit	Player Game	Bond Num games	Johnson a game	Lemieux three games
Ohsumed Dataset				
receive	Patient Therapy	patient chemotherapy	patient adjuvant therapy	children desflurane
undergo	Patient Surgery	NUM patients surgery	NUM patients cholecystectomy	patient radical hysterectomy
have	Patient Disease	NUM patients rash	NUM patients DN (Diabetic Nepropathy)	patient disease

Table 7: Examples of predicates induced by NN-RESCAL from the NYT and Ohsumed datasets. Arguments label are specified by the human annotator, everything else is induced automatically. See Section 4.6 for more details.

convenient starting point for the Predicate Induction problem, a lot of the information in the document can't be captured using such triples. Moreover, in many cases, the knowledge captured in the triples become out of context, thereby making them difficult to interpret even by humans. Augmenting OIE triples with more extractions from the text documents remains an interesting open problem.

Challenge #3: Low recall of all the tensor factorization methods compared remains a concern. Even though predicate invention is a hard problem, improving lower accuracy of the factorization methods is also a challenge.

5 Conclusion

Predicate induction is an important problem as it will help NLU systems. Tensors provide a natural way of representing the triples so we experimented with two main tensor factorization methods and their non-negative variants. Tensor factorization methods have been applied before for the triples from existing Knowledge bases, but no attempts were made in open information extraction settings. We made the first attempt in applying tensor factorization methods for predicate induction from OIE triples and we learned some important insights. We demonstrated with our experiments that non-negativity is a better constraint for interpret-ability and NN RESCAL is best among the methods compared in terms of interpret-ability. We also discovered some open challenges which we would like to pursue as part of our future work.

References

Brett W. Bader, Tamara G. Kolda, et al. 2015. Matlab tensor toolbox version 2.6. Available online, February.

Rasmus Bro and Henk A. L. Kiers. 2003. A new efficient method for determining the number of components in parafac models. *Journal of Chemometrics*, 17(5):274–286.

Nathanael Chambers. 2013. Event schema induction with a probabilistic entity-driven model. In *EMNLP*, pages 1797–1807. ACL.

Kai-Wei Chang, Wen-tau Yih, Bishan Yang, and Christopher Meek. 2014. Typed tensor decomposition of knowledge bases for relation extraction. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1568–1579.

Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 601–610. ACM.

Oren Etzioni, Anthony Fader, Janara Christensen, Stephen Soderland, and Mausam Mausam. 2011. Open information extraction: The second generation. In *IJCAI*, volume 11, pages 3–10.

Luis Galárraga, Geremy Heitz, Kevin Murphy, and Fabian Suchanek. 2014. Canonicalizing Open Knowledge Bases. CIKM.

R. A. Harshman. 1970. Foundations of the PARAFAC procedure: Models and conditions for an" explanatory" multi-modal factor analysis. *UCLA Working Papers in Phonetics*, 16(1):84.

John S. Justeson and Slava M. Katz. 1995. Technical terminology: some linguistic properties and an algorithm for identification in text. *Natural Language Engineering*, 1(1):9–27.

U. Kang, Evangelos Papalexakis, Abhay Harpale, and Christos Faloutsos. 2012. Gigatensor: Scaling tensor analysis up by 100 times - algorithms and discoveries. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, pages 316–324, New York, NY, USA. ACM.

Tamara G Kolda and Brett W Bader. 2009. Tensor decompositions and applications. *SIAM review*, 51(3):455–500.

Denis Krompaß, Maximilian Nickel, Xueyan Jiang, and Volker Tresp. 2013. Non-negative tensor factorization with rescal. *Tensor Methods for Machine Learning, ECML workshop*.

Jey Han Lau, Karl Grieser, David Newman, and Timothy Baldwin. 2011. Automatic labelling of topic models. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 1536–1545, Stroudsburg, PA, USA. Association for Computational Linguistics.

Daniel D. Lee and H. Sebastian Seung. 2000. Algorithms for non-negative matrix factorization. In *In NIPS*, pages 556–562. MIT Press.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60.

T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling. 2015. Never-ending learning. In *Proceedings of AAAI*.

Thahir P. Mohamed, Estevam R. Hruschka, Jr., and Tom M. Mitchell. 2011. Discovering relations between noun categories. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1447–1455, Stroudsburg, PA, USA. Association for Computational Linguistics.

Brian Murphy, Partha Pratim Talukdar, and Tom M Mitchell. 2012. Learning effective and interpretable semantic models using non-negative sparse embedding. In *COLING*, pages 1933–1950.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In Lise Getoor and Tobias Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, ICML '11, pages 809–816, New York, NY, USA, June. ACM.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2012. Factorizing yago: Scalable machine learning for linked data. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, pages 271–280, New York, NY, USA. ACM.

Evangelos E Papalexakis and Christos Faloutsos. 2015. Fast efficient and scalable core consistency diagnostic for the parafac decomposition for big sparse tensors. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*.

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of WWW*.

Paola Velardi, Stefano Faralli, and Roberto Navigli. 2013. Ontolearn reloaded: A graph-based algorithm for taxonomy induction. *Computational Linguistics*, 39(3):665–707.