Feature Learning of Patent Networks Using Tensor Decomposition

Mohamed Maskittou(B), Anass El Haddadi, and Hayat Routaib

Data Sciences and Competitive Intelligence (DSCI) Team, National School of Applied Sciences, Abdelmalek Essaidi University, Al-Hoceima, Morocco
hassanmaskito@gmail.com, anass.elhaddadi@gmail.com, routaib.hayat@gmail.com

Abstract. In the age of big data, the graph clustering algorithms are visual analytics can help the decision-makers to have a precise description of the relationships between data information in the technological areas and companies. Considering the patent as a source of science and technologies this paper presents a new road map for the patent landscape begins with searching and recognizing related text data in similar patent using a deep learning approach to build a model that can exhibit the relationships between text patent in the form of a graph and constructing large patent networks, to complete the last pass in a patent network project and owing to the complexities of the data structure involve the application of the basic tensor concept and their properties to perform automatic unsupervised learning without taking account of the noisy patent data, throw this process the visualization of graphs knowledge much time more powerful and helpful could prove very useful in improving the decision-making process within an organization.

Keywords: Patent analysis · Deep learning · Patent network · Big data analytic · Graph clustering · Node embedding · Tensor decomposition

1 Introduction

Since the beginning of the global public health crisis due to covid-19, people lives changed and become more digital, the adaptation of the new data generation model and data consumption model in different areas health, finance, technology, economy, there are 2.5 Quintilian bytes of data created each day, the world economic forum 2012 confirm that “data is a new class economic asset, like accuracy and gold”, clearly the big data is a massive volume of both structured and unstructured data that require new architecture and techniques to manage it and extract value and hidden knowledge from data.

To overcome the complexity of the data the researchers involved propose innovative methods for the data store and data management, analysis and tools for visualization, one of the useful techniques in the field of visualization techniques is the graphs. We can apply graph technology in a different field and
generate semantic networks like linked open data, social network, patent network, naturally, graphs and data are very linked then big data generate big graphs [1].

Like other structure of data so we can apply analysis on the graphs structure, graph clustering task refuses to cluster data in the form of graphs (nodes, edges) using algorithm inspired by the nature or mathematical models consequently we will use skillfully these models to develop solutions for real-world large scale problems especially if we want to understand how analysing sets of technology documents affect the economic growth and find solutions in crisis.

Patent document has attracted the attention of researchers for its creativity, novelty and practicality. According to the European Patent Office statistics, 80% of all technical information in the world can be found in the patent document. The relevant departments of World Intellectual Property Organization (WIPO) have also done statistics, 90% to 95% of the world’s inventions are published in the form of patent document, of which about 70% inventions have never been published in other non-patent document. If you can apply the patent documents, scientific research institutions can save 40% of R & D expenditures, saving 60% of the time [2]. Patent analysis and patent networks have long been employed as a useful analytical tool for technological opportunity analyses, particularly supporting to create new ideas by using graphs technology and bibliometrics analysis to build the patent network. The semantic patent network helps organizations to identify the technological trends, shows the important competitors, determine the trend shift for ubiquitous technologies in the marketplace [3], by using graphs and patent analysis techniques to build graphs of patent information, exclusively we can accelerate and gain a powerful process of decision making.

The rest of the paper is structured as follows. In Sect.2, we introduce related works with patent analysis and visualisation techniques. In Sect. 3, we explain a new methodology of the clustering patent networks, finally we conclude the paper with a discussion comparing the proposed approaches of clustering graphs based on patent data.

2 Related Works

**Patent Analysis Techniques.** Measuring similarities among bibliometric (journals, patent, authors) is a central task in bibliometric [4], by measuring similarities between the content of the patent (IPC, authors, description, abstract) we can generate a basic set of similar patent, several studies take this challenge, Salton’s cosine [5] and Jaccard’s index [6] are two suitable methods for measuring similarities. Klavans and Boyack [4] compared six approaches for measuring similarity in science maps - raw frequency, cosine, Jaccard’s index, Pearson’s coefficient, average relatedness factor. Boyack performs an other comparative analysis of the accuracy of several text-based approaches to similarity measures, e.g. TF-IDF, latent semantic analysis, and topic models, and from a technical perspective [7], this comparison provides insights for further topic analysis
Feature Learning of Patent Networks Using Tensor Decomposition

The United States Patent and Trademark Office database, and employ a hidden Markov model—which is an unsupervised machine learning technique based on a doubly stochastic process—to estimate the probability of technology being at a certain stage of its life cycle [8], which means deep learning technology is used in these areas [2,9]. Also NLP text mining for patent analysis is popular tool to display the similarities between patent data [10]. Li Shengzhen use BP neural network to classify the patents downloaded from the State Intellectual Property Office network, the category system reaches the main group level of IPC [11], some papers discusses how to apply latent Dirichlet allocation in a trend analysis methodology by exploits patent information [12]. Other work use LDA to visualizing the development paths among patents through sensitivity analyses based on semantic patent similarities and citations [13].

Patent Visualization Techniques. One of the advantages of patent analysis is exploring the outcome of this different techniques to build word graphs, multiple approaches are based on forming a semantic network of keywords from patent documents by clustering the patent document applying Kmeans [3], in addition way, plotting the citation network from kernel k-means clustering using exponential kernel [14], by the semantic network is dependent on the number of groups which is set temporarily by the k-means clustering algorithms, so there are many semantic networks [15], as always the number of clusters have to be determined, mining the optimal number of clusters in a data set is a fundamental issue in partitioning clustering, such as k-means clustering, which requires the user to specify the number of clusters k to be generated [16], therefore, the studies targeted new bag of automatic and nature-inspired algorithms [17,18], other work interest Graph drawing by force-directed [19]. In the era of big data suggest the dimension reduction techniques for the node embedding task, to put it simply, tensors which are a multidimensional extension of matrices a principled and mathematically sound way of modeling such multi aspects’ data [20], in other word graph data modeled as tensors and apply factorization to learn graph embedding, some studies, they work on multi-views graphs [21], faced with huge volume and complexity of data the Mathematical models of basic tensor decomposition to develop algorithms for learning graphs and solve the survey graph embedding problems [22,24].

3 Methodology

Building a patent network is hard work, especially when examining large patent data, the success of this process can be greatly enhanced by the proposed methodology in Fig. 1, thoroughly the process of deep learning and tensor machine learning applications is applied with two different types of data (Textual data, graphs) to display graph clusters based on patent data.

At present contributing practical problem obligate dealing with high dimensional data in many field social network, recommendation system, Aggregation, and Fraud News Detection and technology development as image recognition [25], speech recognition [26], Text classification, or document classification more
specifically are typical Natural Language Processing (NLP) tasks which deal with the automated assignment of one or multiple pre-defined labels to a given text or document respectively [27]. In a recent period, the deep learning algorithms defeat other machine learning algorithms giving significant results and high precision.

3.1 Search Patent and Find Related Terms in Similar Patent

On the way to identify patent topics, the data science methodology of John Rollins is familiarised with the problem-solving concept of the patent analysis as results the dataflow in Fig. 2 is properly manipulated. The U.S. Patents and Trademark Office (USPTO) provides a searchable database of patent applications from the years 2001–present, after, finish the data acquisition from USPTO the preprocessing operations begin, first removing useless information in the document like stop words in the document, there are many useless features in the document, punctuation, numbers, special characters and some other irrelevant information in the documents, second we perform the Word segmentation for documents and divide the sentences in the document into words.

As mentioned in the related works the deep learning enables data scientist to recognize a complex pattern in patent data giving away to technology insight
platforms which are the base of competitive intelligence. In the AI modeling, set up the digital mapping table of vocabulary in the text, statistics on the words, using the text space vector to store, so that the text is expressed as a feature vector [2], besides of building a context-based model for the data pipelines using statistics.

The overview of RBMs (Fig. 3), with a focus on understanding how they work, get started providing a feature patent vector as input to an RBM. The textual data are processed by the input layer RBMs learn patterns and extract important features in data by reconstructing the input. After training is complete, the net can reconstruct the input based on what it learned [28]. The reconstructed patent vector, here, is only a representation of what happens. What’s important to take from this, though, is that the RBM can automatically extract meaningful features from a given input in the training process. A trained RBM can reveal which features are the most important ones when detecting patterns.
It can also represent each patent with some hidden values. Looking at the RBM’s training process, in which, three major steps are repeated. The first step is the forward pass. In the forward pass, the input feature patent vector is converted to binary values, and then, the vector input is fed into the network, where it’s values are multiplied by weights, and overall bias, in each hidden unit. Then, the result goes to an activation function, such as the sigmoid function, which represents the probability of turning each individual hidden-unit on, or in other words, the probability of the node activation. Then, a sample is drawn from this probability distribution, and it finds which neurons may or may not activate. This means it makes stochastic decisions about whether or not to transmit that hidden data. The intuition behind the sampling is that there are some random hidden variables, and by sampling from the hidden layer, you can reproduce sample variants encountered during training. So, as you can see, the forward pass translates the inputs into a set of binary values that get represented in the hidden layer. Then we get to step 2: the backward pass. In the backward pass, the activated neurons in the hidden layer send the results back to the visible layer, where the input will be reconstructed. During this step, the data that is passed backward is also combined with the same weights and overall bias that were used in the forward pass. So, once the information gets to the visible layer, it is in the shape of the probability distribution of the input values, given the hidden values. And sampling the distribution, the input is reconstructed. So, as you can see, the Backward pass is about making guesses about the probability distribution of the original input [29]. Step 3 consists of assessing the quality of the reconstruction by comparing it to the original data. The RBM then calculates the error and adjusts the weights and bias to minimize it. That is, in each epoch, we compute the “error” as a sum of the squared difference between step 1 and the next step. These 3 steps are repeated until the error is deemed sufficiently low [29]. In this section, the basic concept behind autoencoders we’ll be covering. As we increase the dimensionality, the time to deal with data increases exponentially, to train and fit the raw data into a neural network that can detect the patterns. Then extract the most important features of a patent, and represent each patent with those features which are of lower dimensions. An autoencoder works well for this type of problem. An autoencoder is a type of unsupervised learning algorithm that will find patterns in a dataset. Generally speaking, autoencoders excel in tasks that involve feature learning or extraction, data compression, and learning generative models of data and dimensionality reduction [29]. A deep belief network is a kind of deep learning network formed by stacking several RBMs. The loss function of the sparse encoder is, sparse limit quantitative KL divergence and sparse penalty factor, which can effectively control the sparsity:

\[
J_{spr}(W, B) = J(W, b) + \beta \sum_{j=1}^{S_2} KL(p||p_j -) \tag{1}
\]

\[
KL(p||p_j) = plg \frac{p}{p_j -} + (1 - p)lg \frac{1 - p}{1 - p_j} \tag{2}
\]
Feature Learning of Patent Networks Using Tensor Decomposition

to explain the back propagation algorithm. Suppose a fixed training set
\((x^1, y^1), ..., (x^m, y^m)\) of \(m\) training examples. For a single training example \((x, y)\),
the loss function will respect that single example to be:

\[
J(W, b; x, y) = \frac{1}{2} ||J_W(b)(x) - y||^2
\]  

(3)

The error of the neurons node in the input layer is:

\[
\delta_i^{(3)} = \frac{\partial}{\partial J(W, b; x, y)} = -(y_i - a_i^{(3)}) f'(z_i^3)
\]  

(4)

the partial derivative is

\[
\frac{\partial}{\partial w_{i}^{(l)}} j(W, b; x, y) = \delta_i^{(l+1)} a_j^{l}
\]  

(5)

\(x\): Input features for a training example.
\(\theta\): Parameter vector. It is useful to think of this as the result of taking the parameters \(W, b\) and “unrolling” them into a long column vector.
\(f(.):\) The activation function.
\(z_i^{(l)}\): Total weighted sum of inputs to unit \(i\) in layer \(l\).

The Softmax regression is used as an activation function in classification, it generates the probabilities for the output, and the features learned from the feature selection module being used as the input of Softmax classifier

\[
\begin{bmatrix}
P(y^1) \\
P(y^2) \\
\vdots \\
P(y^k)
\end{bmatrix} = \frac{1}{\sum_{j=1}^{k} e^{\theta_j^T x^{(i)}}} \times
\begin{bmatrix}
e^{\theta_1^T x^{(i)}} \\
e^{\theta_2^T x^{(i)}} \\
\vdots \\
e^{\theta_k^T x^{(i)}}
\end{bmatrix}
\]

3.2 Plot Patent Network or Graphs Information

Based on the results of the previous stage. For the visualization GraphX is a new component in Spark for graphs and graph-parallel computation. At a high level, GraphX extends the Spark RDD by introducing a new Graph abstraction: a directed multigraph with properties attached to each vertex and edge. To support graph computation, GraphX exposes a set of fundamental operators (e.g., subgraph, joinVertices, and aggregate Messages) as well as an optimized variant of the Pregel API. In addition, GraphX includes a growing collection of graph algorithms and builders to simplify graph analytics tasks [41].

3.3 Clustering Graphs Data

Graphs or patent networks are a general language for describing and modeling complex relationships between patents we carried on the work to deal with
the high dimensional data to represent this complex interconnected communication system in the same time we extract small clusters which are core patent sets consist valuable information, early the researchers tend to use classical MI algorithms in graphs: node classification, link prediction, community detection, network similarity. Graphs are far more complex than text or visual data, the nodes haven’t a fixed node ordering or reference due to the isomorphism problem, often dynamic and have multimodal features. Nowadays, the studies tend to be more concerned with the similarities between nodes, by using nodes embedding algorithms to learn low dimensional vector representations for nodes in a graph. To run the process of learning embedding nodes, it’s necessary owning an embedding space generated by optimizing the parameters of the node similarity function. Nowadays, there are three famous unsupervised feature learning. Matrix decomposition-based approaches, multihop similarity-based approaches, and random walk-based approach [31].

Matrix decomposition-based approaches decompose various matrix representations of graphs by eigendecomposition or Singular Value Decomposition (SVD) [22]. The similarity function is just the edge weight between u and v in the original network, the intuition is Dot products between node embeddings approximate edge existence [31].

\[
\gamma = \sum_{(u,v)\in(U,V)} \| Z_u^T Z(v) - A_{(u,v)} \|^2
\]  
(6)

loss function:

\[
\gamma
\]

Sum over all pairs:

\[
\sum_{(u,v)\in(U,V)}
\]

Embedding similarity:

\[
Z_u^T Z(v)
\]

adjacency matrix for the graph (weighted):

\[
A_{(u,v)}
\]

The goal is finding an embedding matrix using stochastic gradient descent (SGD) or Solve matrix decomposition solvers, this approach has drawbacks like considering all nodes pairs, and the ability to learn one node per vector [31].

The basic idea behind the multi-hop similarity is train embedding to predict k-hop neighbors. By this similarity function.

\[
\gamma = \sum_{(u,v)\in(U,V)} \| Z_u^T Z(u,v) - A_{(u,v)}^k \|^2
\]  
(7)

The measure of overlap between nodes neighborhoods in this case by practicing log-transformed, (probabilistic adjacency matrix) and train multiple different
hop lengths and concatenate output [32]. The other way by overlap function like Jaccard similarity [33].

\[
\gamma = \sum_{(u,v) \in (U,V)} \|Z^T_{u} Z_{(u,v)} - S_{(u,v)}\|^2
\]  

(8)

loss function:

\[
\gamma = \sum_{(u,v) \in (U,V)} \|Z^T_{u} Z_{(u,v)} - S_{(u,v)}\|^2
\]

Sum over all pairs:

\[
\sum_{(u,v) \in (U,V)} \|Z^T_{u} Z_{(u,v)} - S_{(u,v)}\|^2
\]

Embedding similarity:

\[
Z^T_{u} Z_{(u,v)}
\]

The neighborhood overlap between u and v (e.g., Jaccard overlap)

\[
S_{(u,v)}
\]

This method has also some drawbacks, generally need to iterate over all pairs of nodes that make the process expensive and time consuming beside the dealing with the massive parameter space.

Concisely, the previous methods show features that are not easily interpretable, to rise this challenge and generate an interpretable feature space for the nodes, tensor decomposition-based node embedding algorithms can manage the situation of the nodes with high accuracy.

The review papers [34], tensors are multidimensional arrays. In the proposed method of node embedding using tensor decomposition, considering the third-order tensors and CP decomposition Fig. 4.

![Fig. 4. CP decomposition of a third-order tensor. [30]](image)

briefly review the CP decomposition, CP decomposition factorizes the tensor into a sum of rank-one tensors. Given a third-order tensor \(X \in R^{I \times J \times K}\), where I, J, and K denote the indices of tensor elements in three of its modes, CP decomposition factorizes the tensor in the following way.

\[
X \approx \sum_{(r=1)}^{R} a_r o b_r o c_r = [A, B, C]
\]  

(9)
Here, $\circ$ denotes the outer product of the vectors, $R$ is the tensor rank which is a positive integer, $a_r$, $b_r$, and $c_r$ are vectors, where $a_r \in (R^I)$, $b_r \in (R^K)$, and $c_r \in (R^K)$ for $r = 1, 2, 3, ..., R$. After stacking those vectors, we can get the factor matrices $A = [a_1, a_2, ..., a_R]$, $B = [b_1, b_2, ..., b_R]$, and $C = [c_1, c_2, ..., c_R]$, where $A \in R^{I \times R}$, $B \in R^{J \times R}$, and $C \in R^{K \times R}$. Figure 5 is a visualization of the CP decomposition of a third-order tensor. The matricized forms of the tensor $X$ is given by,

$$X(1) \approx A(C \otimes B)^T, X(2) \approx B(C \otimes A)^T, X(3) \approx C(B \otimes A)^T$$

(10)

where $\otimes$ represents Khatri-Rao product of two matrices. ALS Solution of CP Decomposition: CP decomposition can be solved by Alternating Least Squares [34]. The cost function of CP decomposition can be formulated as,

$$\min_{A,B,C} \|X - \sum_{(r=1)}^{R} a_r b_r c_r\|_F^2$$

(11)

where the $\|\cdot\|_F$ the tensor Frobenius norm which is the sum of squares of all elements of the tensor. By initializing $B$ and $C$ with random values, ALS updates $A$ by following rule.

$$A \leftarrow \arg\min_A \|X(1) - A(C \otimes B)^T\|_F^2$$

(12)

Then by xing $A$ and $C$, it updates $B$ by

$$B \leftarrow \arg\min_A \|X(2) - B(C \otimes A)^T\|_F^2$$

(13)

Then by xing $A$ and $B$, it updates $C$ by

$$C \leftarrow \arg\min_C \|X(3) - C(B \otimes A)^T\|_F^2$$

(14)

the last three equations are repeated until convergence of ALS solution of CP decomposition.

**Fig. 5.** CP decomposition-based representation learning of source nodes, target nodes, and transition steps [22]

The algorithms of tensor decomposition based node embedding use CP decomposition to extract factor matrices containing the representations of the source and/or target properties of the nodes, and the transition steps.
After we find the source factor matrix $A$, target factor matrix $B$, and transition factor matrix $C$, we can compute the projection of source embedding of node $i$ on the transition embedding $j$, where $1 \leq i \leq n$ and $1 \leq j \leq K$, and get source-transition embedding matrix $ST \in R^{n \times K}$. Similarly, we can get a target-transition embedding matrix $TT \in R^{n \times K}$ that reflects the projection of target embeddings on transition step embeddings. Finally, we get the node embedding matrix $Z \in R^{n \times 2K}$ by concatenating $ST$ and $TT$. First $K$ columns of $Z$ represent source role of a node with varying transition steps, and last $K$ columns of $Z$ represent target role of a node with varying transition steps [22].

\[
ST = A * CT \tag{15}
\]
\[
TT = B * CT \tag{16}
\]
\[
Z = [ST, TT] \tag{17}
\]

the Case TDNEperSlice:

\[
ST^k = A^{(k)} \times C^{(k)T} \tag{18}
\]
\[
(TT)^k = B^{(k)} \times C^{(k)T} \tag{19}
\]
\[
Z = [ST^{(1)}, ST^{(2)}, ...ST^{(K)}, TT^{(1)}, TT^{2}, ...TT^{K}] \tag{20}
\]

| Input: 1-step transition probability matrix $A$ |
| Maximum transition step $K$ |
| CP decomposition rank $R$ |
| Output: Node embedding matrix $Z$ |

Source-transition embedding matrix $ST$

1: $n = \text{count rows}(A)$
2: for $k$ in 1 to $K$ do
3: $X^k = \text{tensor}(n, n, 1)$
4: $X^k = A^k$
5: $[A^k, B^k, C^k] \leftarrow \text{CPALS}(X^k, R)$
6: $(ST)^k = A^k \times (C^k)^T$
7: $TT^k = B^k \times (C^k)^T$
8: end for
9: $Z = [ST^{(1)}, ST^{(2)}, ...ST^{(K)}, TT^{(1)}, TT^{2}, ...TT^{K}]$
10: return $Z$

Algorithm TDNE: Tensor Decomposition-based Node Embedding
Input: 1-step transition probability matrix $A$
Maximum transition step $K$
Source-transition embedding matrix $ST$
CP decomposition rank $R$
Output: Node embedding matrix $Z$

1: $n = \text{count rows}(A)$
2: for $k$ in 1 to $K$ do
3: for $k$ in 1 to $K$ do
4: $X(:, :, k) = A^k$
5: end for
6: $[A, B, C] \leftarrow \text{CPALS}(X, R)$
7: $ST = A \times C^T$
8: $Z = [ST, TT]$ 9: $Z = [ST, TT]$
10: return $Z$

Algorithm TDNE per: Tensor Decomposition-based Node Embedding

3.4 Visualization and Evaluation

This stage provides tips to evaluate the model of tensor decomposition based node embedding and carry out a comparative study with other models for nodes embedding, Laplacian Eigenmaps (LAP) [35], LLE [36], HOPE [37], GraRep [38], DeepWalk [39], Node2vec [39], GraRep [41] using different dataset brain network, social network, Air traffic network [22]. To reach the accuracy and precision in network reconstruction, and Precision in link prediction (Figs. 6, 7 and 8).

| Baseline algorithms | Parameters |
|---------------------|------------|
| GRAPrep             | Maximum transition step, and log shifted factor |
| DeepWalk            | Walks per node, walk length, context size, return parameter, in-out parameter |
| HOPE                | The decay parameter for Katz Index transition step, and log shifted factor $\beta$ |
| Node2Vec            | Walks per node, walk length, context size, return parameter, in-out parameter |
Fig. 6. Precision in link prediction [22]

Fig. 7. Precision in network reconstruction [22]

Fig. 8. Precision in network reconstruction [22]
4 Conclusion

In this paper, a new methodology to build patent networks using patent document classifier based on deep learning for automatic classification to enhance high performance for the patent analysis stage, to do so in another stage of the methodology a tensor decomposition and higher-order transition probability matrices to achieve feature learning of Graphs, as a result, higher accuracy and precision compared to bases algorithms, consequently, patent networks allow analysts to visualize interconnected patent and leverage their visual abilities to decipher knowledge of a important and could prove very useful in improving the decision-making process within an organization.

References

1. Kheddouci, H.: Big Data et Graphes: Défis et pistes de recherche. French Laboratoire d’InforMatique en Image et Systèmes d’information LIRIS UMR 5205 CNRS/INSA de Lyon/Université Claude Bernard Lyon 1/Université Lumière Lyon 2/Ecole Centrale de Lyon
2. Xia, B., Zur Elektrodynamik, Li, B., Lv, X.: 2nd International Conference on Artificial Intelligence and Industrial Engineering (AIIE 2016), Advances in Intelligent Systems Research. China Research on Patent Document Classification Based on Deep Learning, vol. 133, pp. 308–311 (2016)
3. Kim, Y.G., Suh, J.H., Park, S.C.: Visualization of patent analysis for emerging technology (South Korea). Expert Syst. Appl. 34, 1804–1812 (2008)
4. Zhang, Y., Shang, L., Huang, L., Porter, A.L., Zhang, G., Lu, J., Zhu, D.: A hybrid similarity measure method for patent portfolio analysis. J. Infomatr. 10, 1108–1130 (2016)
5. Salton, G., Buckley, C.: Term-weighting approaches in automatic text retrieval. Inf. Process. Manag. 24(5), 513–523 (1988). J. Am. Soc. Inf. Sci. Technol. 57
6. Klavans, R., Boyack, K.W.: Identifying a better measure of relatedness for mapping science, USA, vol. 57, no. 10, pp. 251–361 (2006)
7. Boyack, K.W., Newman, D., Duhon, R.J., Klavans, R., Patek, M., Biberstine, J.R., Börner, K.: Clustering more than two million biomedical publications: comparing the accuracies of nine text-based similarity approaches. PLoS One 6(3), 53–64 (2016). J. Am. Soc. Inf. Sci. Technol. 57
8. Lee, C., Kim, J., Kwon, O., Woo, H.-G.: Stochastic technology life cycle analysis using multiple patent indicators. J. Am. Soc. Inf. Sci. Technol. 106, 53–64 (2016)
9. Li, S., Hu, J., Cui, Y., Hu, J.: DeepPatent: patent classification with convolutional neural networks and word embedding. J. Am. Soc. Inf. Sci. Technol. 117, 721–744 (2018)
10. Abbas, A., Zhang, L., Khan, S.U.: A literature review on the state-of-the-art in patent analysis. Word Patent Inf. 37, 3–13 (2018)
11. Li, S., Wang, J., Qu, J.: Automated, categorization of patent based on back-propagation network. Comput. Eng. Des. 31(25), 5075–5078 (2010)
12. Oh, S., Choi, S., Yoon, J., Choi, H.: Innovation topic analysis of technology: the case of augmented reality patents. J. Am. Soc. Inf. Sci. Technol. USA (2018)
13. Kim, G., Park, S., Jang, D.: Technology analysis from patent data using latent Dirichlet allocation. Soft Comput. Big Data Process. 57(10), 71–80 (2014)
14. Kim, D., Lee, B., Lee, H.J., Lee, S.P., Moon, Y., Jeong, M.K.: Graph Kernel approach for detecting core patents and patent groups. In: AIP Conference Proceedings, vol. 1827 (2014)
15. Suh, J.H., Park, S.C.: A new visualization method for patent map: application to ubiquitous computing technology. In: Advanced Data Mining and Applications, pp. 566–573 (2006)
16. Shanie, T., Suprijadi, J., Zulhanif: Determining The Optimal Number Of Clusters: 3 Must Know Methods, Cluster Validation Essentials (2020)
17. Kennedy, J., Eberhart, R.C.: Swarm intelligence. Scholarpedia 2, 1462 (2020)
18. Boulouard, Z., El Haddadi, A., Dousset, B.: “Forced” Force Directed Placement: a New Algorithm for Large Graph Visualization (2018)
19. Thomas, M., Fruchterman, J., Edward Reingold, M.: Graph Drawing by Force-directed Placement, Department of Computer Science, University of Illinois at Urbana-Champaign, USA
20. Papalexakis, E.E., Faloutsos, C.: Unsupervised tensor mining for big data practitioners. Big Data 4(3), 179–191 (2016)
21. Wu, J., Xie, X., Nie, L., Lin, Z., Zha, H.: Unified graph and low-rank tensor learning for multi-view clustering. Association for the Advancement of Artificial Intelligence (www.aaai.org) (2020)
22. Hamidi, S.M., Angryk, R.: Interpretable feature learning of graphs using tensor decomposition. In: 2019 IEEE International Conference on Data Mining (ICDM) (2020)
23. Bailly, R., Rabusseau, G.: Graph Learning as a Tensor Factorization Problem (2017)
24. Malik, O.A., Ubaru, S., Horesh, L., Kilmer, M.E., Avron, H.: Tensor Graph Convolutional Networks for Prediction on Dynamic Graphs (2020)
25. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1090–1098 (2012)
26. Mikolov, T., Deoras, A., Povey, D., Burget, L., Cernocky, J.: Strategies for training large scale neural network language models. In: Automatic Speech Recognition and Understanding, pp. 196–201 (2011)
27. Basili, R., Moschitti, A., Pazienza, M.T.: NLP-driven IR: evaluating performances over a text classification task, University of Rome Tor Vergata Department of Computer Science, Systems and Production 00133 Roma (Italy) (2020)
28. Nayak, M.: Dimensionality Reduction and Feature Extraction with RBM. www.medium.com/tag/deep-learning/latest (2020)
29. Akelson, A.: Restricted Boltzmann Machines (RBMs) (2020). www.coursera.org/lecture/building-deep-learning-models-with-tensorflow
30. Navamani, T.M.: Efficient deep learning approaches for health informatics (chap. 7). In: Deep Learning and Parallel Computing Environment for Bioengineering Systems, pp. 123–137 (2019)
31. Ahmed, A., Shervashidze, N., Narayanamurthy, S., Josifovski, V., Smola, A.J.: Representation learning on graphs: methods and applications. IEEE Data Eng. Bull. Graph Syst. (2018)
32. Cao, S., Xu, Q., Lu, W.: GraRep: learning graph representations with global structural information. In: Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, pp. 891–900, October 2015
33. Ou, M., Cui, P., Pei, J., Zhang, Z., Zhu, W.: Asymmetric transitivity preserving graph embedding. In: The International World Wide Web Conference Committee (IW3C2), pp. 13–17 (2016)
34. Rabanser, S., Shchur, O., Günnemann, S.: arXiv:1711.10781v1. [stat.ML] (2017)
35. Belkin, M., Niyogi, P.: Laplacian eigenmaps and spectral techniques for embedding and clustering. In: Advances in Neural Information Processing Systems, NIPS 2001, Vancouver, British Columbia, Canada, 3–8 December 2001, pp. 585–591 (2001)
36. Roweis, S., Saul, L.: Nonlinear dimensionality reduction by locally linear embedding. Science 290(5500), 2323–2326 (2000)
37. Ou, M., Cui, P., Pei, J., Zhang, Z., Zhu, W.: Asymmetric transitivity preserving graph embedding. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016, pp. 1105–1114 (2016)
38. Cao, S., Lu, W., Xu, Q.: GraRep: learning graph representations with global structural information. In: Proceedings of the 24th ACM International Conference on Information and Knowledge Management, CIKM 2015, Melbourne, VIC, Australia, 19–23 October 2015, pp. 891–900 (2015)
39. Grover, A., Leskovec, J.: node2vec: scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016, pp. 855–864 (2016)
40. Perozzi, B., Al-Rfou, R., Skiena, S.: DeepWalk: online learning of social representations. In: The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2014, New York, NY, USA, 24–27 August 2014, pp. 701–710 (2014)
41. https://spark.apache.org/docs/latest/graphx-programming-guide.html