International Journal of Energy, Information and Communications
Vol.7, Issue 5-6 (2016), pp.13-20
http://dx.doi.org/10.14257/ijeic.2016.7.6.02

Visualization of Graph Models for Web Document in Neo4j

Surajit Medhi

Department of Computer Science,
Gauhati University, Guwahati-14, Assam, India
surajitmdh@gmail.com

Abstract

The goal of this article is to visualize the graph models for web document in a graph database called neo4j. There are different types of graph models to represent the web document. We represent the different graph models in neo4j using cypher query language. The main purpose of the graph models is clustering the web documents. So, first we have to convert the web document into a graph model. A graph can represent any document with minimum loss of information.

Keywords*: Graph, neo4j, tag, context, cypher query language*

1. Introduction

The exponential growths of the amount of content on the Internet also need to manage the document. So, these web documents are representing using some graph models with minimum loss of information. These graph models are designed mainly for clustering the web documents.

There are mainly three distinct categories for web document clustering [1]:

a) Based on Content

b) Based on Usage

c) Based on Structure

We need to study the actual content of web pages and then apply some method to learn about the pages in clustering based on web content. Generally this is done to organize a group of documents into related categories. This is beneficial for web search engines, since it allows users to more quickly find the information they are looking for in comparison to the usual infinite ordered list.

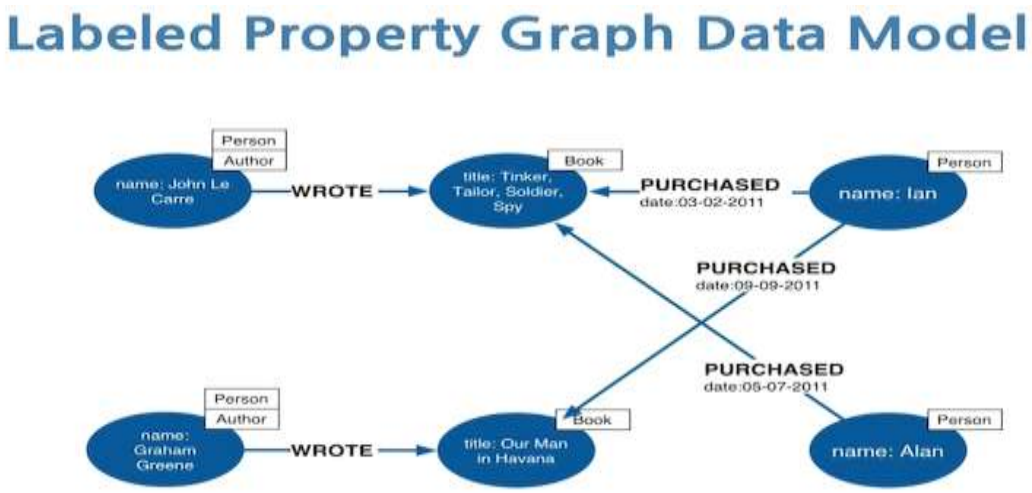
The goal is to examine web page usage patterns in order to learn about a web system's users or the relationships between the documents on the basis of association rules created from web access logs which store the identity of pages accessed by users along with other information such as when the pages were accessed and by whom.

We need to examine only the relationships between web documents by utilizing the information conveyed by each document's hyperlinks in the third category of web document clustering. In this article, we are going to discuss only the graph models for representing the web content in neo4j. Chowdhury *et.al.* [6] are visualizing wikipedia using a Graph Database. They also used Neo4j and cypher query language. Chuntao Jiang *et.al.* [10] used graph mining-based feature extraction technique for text classification. So that graph is an important part of the mining field and also the graph database is important to represent the more linked data.

ISSN: 2093-9655 IJEIC

2. Neo4j

Neo4j is the world's leading open source Graph Database that has a labeled property



graph model. A labeled property graph model has a number of characteristics. It contains node and relationship. Nodes contain properties (key-value pairs) and it can be labeled with one or more labels. In this model, relationships are named and directed, and always have a start node and end node. Relationships can also contain properties. The above graph model is from the neo4j website [2].

This is one example of a graph model represented in Neo4j. Neo4j is written in Java language. Other Graph Databases are Oracle NoSQL Database, Orient DB, Hyper Graph DB, Graph Base, Infinite Graph, and Allegro Graph. But in this article, we select neo4j to represent the web document. Neo4j supports a lot of nodes, relationships and labels. The capacity of nodes, relationships and labels are around 35 billion, around 35 billion and around 75 billion respectively [3]. Neo4j supports ACID rules and also it provides REST API to be accessed by any Programming Language like Java, Spring, Scala etc. It provides Java Script to be accessed by any UI MVC Framework like Node JS. It also supports two kinds of Java API cypher API and Native Java API to develop Java applications. We can easily download Neo4j from the website of neo4j. It uses cypher query language to retrieve information from the database. According to the Neo4j website, Neo4j is an embedded, disk-based, fully transactional Java persistence engine that stores data structured in graphs rather than in tables. Apache's Lucene 3 is used to build Neo4j for indexing and search. Lucene is a text search engine, written in Java, geared toward high performance [4]. As we compared Neo4j with relational database the more link data are smoothly represented in the graph database.

Merits of Neo4j: The representation of connected data is very easy and also easy and faster to retrieve or traversal or navigation of more Connected data. It uses simple and powerful data model. Neo4j cypher query language commands are in humane readable format and very easy to learn and also does not require complex joins to retrieve connected or related data.

Demerits of Neo4j: In the latest version of Neo4j, it has a limitation of supporting the number of Nodes, Relationships and Properties.

3. The Cypher Query Language

The Cypher Query Language is a query language which is used in the Neo4j graph database and also it is a declarative pattern matching language. Neo4j provides a query language cypher[3] which is used to populate and query the database It follows the SQL

like syntax but many differences are there also. Its syntax is very simple and of human readable format. Like SQL, Neo4j also supports different types of clauses like where, order by *etc.*, and also support string and aggregate function. There are lots of commands used in cypher query language, for example create, match return etc. It allows us to describe what we want to select, insert, update or delete from a graph database without requiring us to describe exactly how to do it. Suppose 'a' and 'b' are two nodes and a relation between two nodes is 'like' then the cypher is: (a) – [: like] -> (b). Cypher uses ASCII-Art to represent patterns. We surround nodes with parentheses which look like circles, *e.g.*, **(node)**. If we later want to refer to the node, we shall give it a variable like **(p)** for person or **(t)** for thing. In real-world queries, we'll probably use longer, more expressive variable names like **(person)** or **(thing)**. If the node is not relevant to your question, you can also use empty parentheses **()**. To fully utilize the power of our graph database we want to express more complex patterns between our nodes. Relationships are basically an arrow --> between two nodes. Additional information can be placed in square brackets inside of the arrow [4].

4. Graph Models for Web Document Representation

There are several methods for representing web document content or text documents in general as graphs. These methods are named: standard, simple, n- distance, n-simple distance, absolute frequency and relative frequency. Schener *et.al.* [1] developed the standard, simple, n- distance and n-simple distance, methods to represent the web document as a graph. They do not model the entire document as a graph when creating the graph model of a web document. Some pre-processing steps are performed to arrive at a reduced set of the most important terms by them. They remove some stop words such as 'the', 'and' and 'of' *etc*. and also perform some simple stemming in order to determine those word forms which should be considered to be identical *e.g.* "come" and "comes".

Under the standard representation or Tag Sensitive Graph Model, each term after stop word removal and stemming becomes a vertex in the graph representing that document. Each node is labeled with the term it represents. The node labels are unique in a document graph, since a single node is created for each keyword even if a term appears more than once in the text. Second, if word *aaa* immediately precedes word *bbb* somewhere in a "section" *s* of the document, then there is a directed edge from the node corresponding to term *aaa* to the node corresponding to term *bbb* with an edge label *s*. An edge is not created between two words if they are separated by certain punctuation marks such as periods. They have defined sections for HTML documents are: *title,* which contains the title's text related to the document and link, which is clickable hyper-links on the document that is also a text; and text, which comprises any of the readable text in the body part, and this includes link text but not title and keyword text. According to this standard method we are going to design the graph model in Neo4j.

The third type of representation is called the *n- distance representation* or Context Sensitive Graph Model. In this method, there is a user-provided parameter, *n*. Instead of considering only terms immediately following a given term in a web document, we look up to *n* terms ahead and connect the succeeding terms with an edge that is labelled with the distance between them (unless the words are separated by certain punctuation marks). For example, if we had the following text on a web page, "AAA BBB CCC DDD", then we would have an edge from term AAA to term BBB labelled with a 1, an edge from term AAA to term CCC labelled 2, and so on.

Recently Phukon [7] developed a composite graph model for representing web documents based on the above two models. The composite method of web document representation takes into account additional web-related content information which is not done in traditional information retrieval models. All the necessary information such as the order, proximity of occurrence, mark-up information and location of a word within a

document can hold in this method. To represent three sections namely head, link and address in TSGM because these three sections are comparatively much smaller than the text section and CSGM is capable of representing large section more efficiently than that of TSGM. Use of CSGM will not be possible to utilize the mark-up information available but TSGM will enable.

Quynh Do[10] designed a graph model for text analysis and text mining. They also compared the vector space model with graph model and get a better result in order to store the text.

5. Experiments and Results

Our experiment was performed on some simple web document; we have designed the above mentioned graph models using the cypher query language in Neo4j graph database. For example, we have a simple web document which contains the title "Tezpur University", a link whose text reads "Other Universities in Assam", and text containing "Tezpur University Secures 5th In All India Ranking". According to TSGM, we have designed the graph model in Neo4j graph database.

According to CSGM, for example, if we had the following text on a web page, "EEE FFF GGG HHH", then we would have an edge from term EEE to term FFF labelled 1, an edge from term EEE to term GGG labelled 2, and so on. CSGM also implemented in Neo4j.The user provided parameter n=3 in this web page.

According to the composite graph model, we also have designed the graph model in Neo4j. For example, we had the following web document which contains title "Gauhati University", a link whose text reads "Other Universities In Assam", an address that contain "Powered by xyz" and text containing "Gauhati University Secures 26th In All India Ranking". Under this model title, link and address are represented using TSGM and text means inside the body tag are represented using CSGM. The user provided parameter n=2 in this example.

In the following figures we have seen that the representation of web document in a graphical manner with the help of Neo4j. But in TSGM and CSGM the web documents are represented using the graphical manner with the help of some programming languages such as C++, java etc. In Neo4j we can visualize the graph models with the help of cypher query language and also we can use the API also. Suppose if we want to see the content of the web document then write a query in the Neo4j and show the result in a graphical style as a node and also we know that the next word which is connected by an edge. The nodes and edges are created by using the cypher query language.

The results are shown below-

Figure 1 is the representation of a simple web page using the TSGM in Neo4j. We create the nodes and the relationship using the cypher query language.

Figure 2 is the representation of a simple web page using the CSGM in Neo4j. In this model we represent only the text part using a distance between the nodes and also there is a user provided parameter n=3.In this model the tag are not included so that we don't know that which node is head or title or link.

Figure 3 is the representation of a simple web page using the composite graph model.

This model is the combination of the TSGM and CSGM. The user provided parameter n=2. In this model there are lots of advantages because if we combine the both models then we extract more information.

5.1. Representation of TSGM in Neo4j

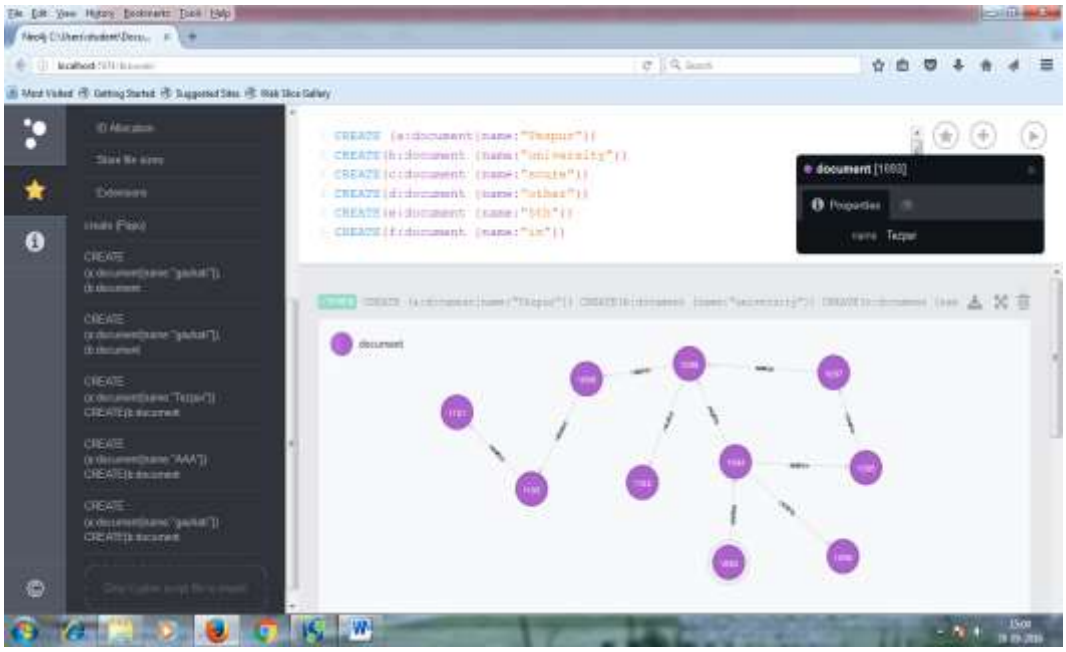


Figure 1. Representation of Web Document using TSGM in Neo4j

In the above figure, we have represented a simple web page using the TSGM in Neo4j. There are different nodes for representing the content in the web page which consist a single word in the document. If the same word is repeated in the page then only one node is created for this particular word. As we see in the above figure, for representing the web content, first we have created the node for every word which is present in the web page and also we have created a relation among the nodes. The relations are actually the tags. According to the TSGM, if the word *aaa* immediately precedes the word *bbb* somewhere in a "section" *s* of the document, then there is a directed edge from the node corresponding to the term *aaa* to the node corresponding to the term *bbb* with an edge label *s*. We know that Neo4j has a labelled property Graph Model. Every node has a label and number of properties. In our example, node label is the document and the properties are contents in the web pages. Every word is represented as a node and the label is called document. Relationships are represented by the tags like title, link, text etc. In our example, we named the relationship as relation and the properties of the relation are the tags.

CREATE (a: document {name:"Tezpur"}), using this command we have created the node Tezpur and the label is document. CREATE (a)-[:relation {roles: “head"}]->(b), using this command we are created the relation between two nodes which are ladled by ‘a’ and ‘b’ and the properties of relation are ‘head’ and also we can put more than one properties for a relation. As we see in the above figure we can also know that which node is represented the particular word by clicking the node in the Neo4j and also know that the relation which is represented by some tags. In that way we have represented a simple web page using cypher query language in Neo4j. From the above model which is designed in Neo4j we can easily find the content in the web page and also the relationship types by clicking the node and relation. Also we can easily search the word which connected to another word and which tag is associated with between these two nodes by using some simple cypher query command. In Neo4j graph database we can create millions of node and relationship

5.2. Representation of CSGM in Neo4j

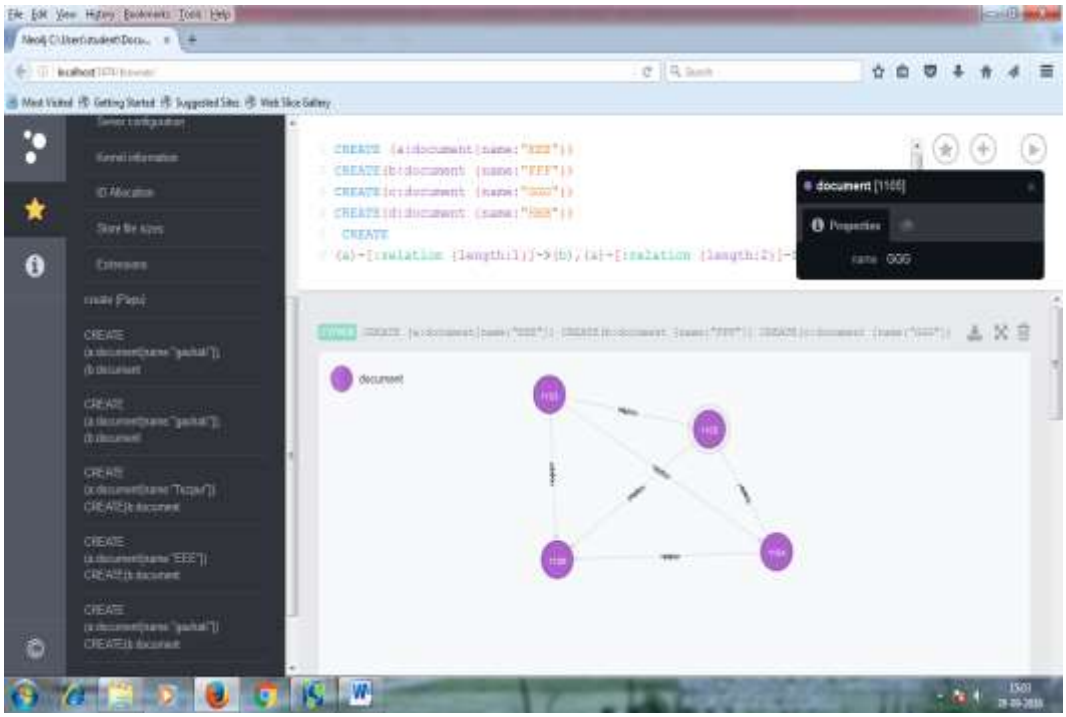


Figure 2. Representation of Web Document using CSGM in Neo4j

In the above figure, we have represented the simple web page using the CSGM in Neo4j. In this model, we create the node as a content and the relations are distance between the nodes and user provided parameter n=3. In the above figure we represented a simple example but we can represent a large document using that model.

5.3. Representation of Composite Graph Model in Neo4j

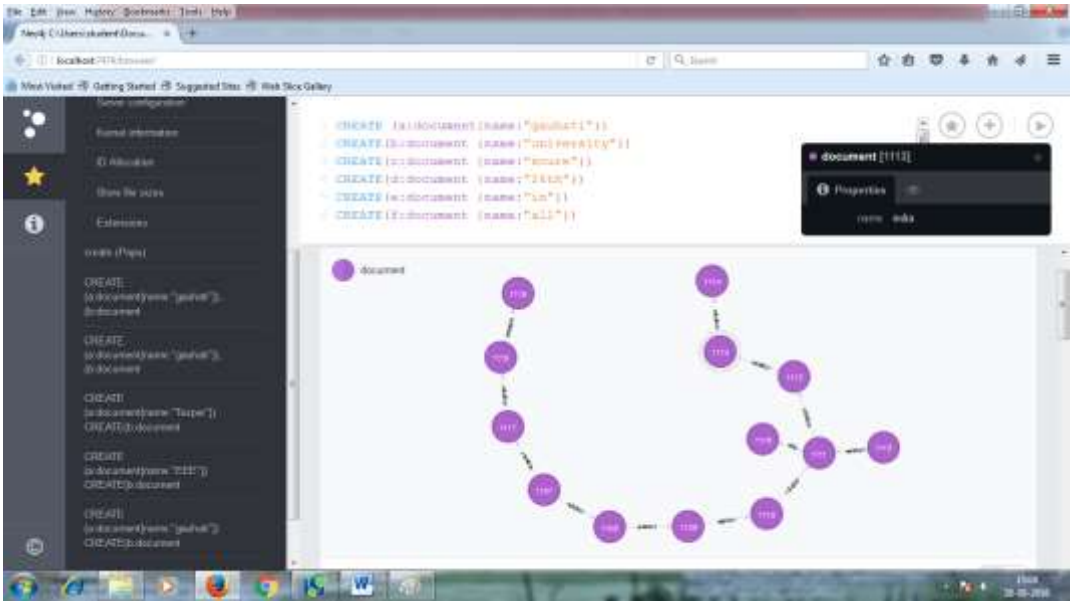


Figure 3. Representation of Web Document using Composite Graph Model in Neo4j

Under the composite graph model we represented the web document using the both TSGM and CSGM. The content of head, link, address and title tags are represented by the TSGM and the text parts that means the under body tag are represented by CSGM. In this representation we can create multiple properties of a relation that means both the tag and the distances. In our example, user provided parameter n=2.

6. Conclusions

The Neo4j graph database is a useful database server. In this article, we have used this server to represent different graph models for simple web documents. We have used the cypher query language to represent the graph models in Neo4j. Also we have used REST API for representing the graph models in Neo4j. If we compare the Neo4j to MySQL, then there are many advantages over MySQL. So we represented three different graph models in Neo4j.Also the web document is represented using the vector space model and using the language like C++, java *etc.*, but don't visualize that content and even the graph models that we are mentioned using the adjacency matrix and adjacency list to represent the web document. So using the graph database *i.e.*, Neo4j we can represent the content and also visualize that content.

References

[1] H. Bunke Schener, M. Last and A. Kandel, "Graph Theoretic Techniques for Web Content Mining", Series in Machine Perception and Artificial Intelligence, World Scientific Publishing Co. Pte. Ltd., vol. 62, **(2005)**.
[2] www.neo4j.com.
[3] "Cypher" - http://neo4j.com/developer/cypher.
[4] www.tutorials.com.
[5] C. Vicknair, M. Macias, Z. Zhao, X. Nan, Y. Chen and D. Wilkins, "A Comparison of a Graph Database and a Relational Database", ACMSE '10, Oxford, MS, USA, **(2010)** April 15-17.
[6] R. Chowdhury, L. Hagberg, J. Sievers and H. Nyblom, "Visualizing Wikipedia using a Graph Database", https://www.kth.se/social/files/.
[7] K. K. Phukon, "A Composite Graph Model for Web Document and the MCS Technique", International Journal of Multimedia and Ubiquitous Engineering, vol. 7, no. 1, **(2012)** January.
[8] N. Deo, "Graph Theory with Applications to Engineering and Computer Science", PHI Learning Private Limited. PHI Learning Private Limited.ISBN-978-81-203-0145-0.
[9] J. J. Miller, "Graph Database Applications and Concepts with Neo4j", Proceedings of the Southern Association for Information Systems Conference, Atlanta, GA, USA March 23rd-24th, **(2013)**.
[10] T. Ngoc Quynh Do, "A Graph Model for Text Analysis and Text Mining, master thesis", erasmus mundus masters program in language and communication technology, **(2012)** June 18.

Authors

Surajit Medhi, Guest Faculty, Department of Computer science, Gauhati University, India received MSc(IT) degree from Gauhati university in 2011. He is currently pursuing doctoral research at Department of Computer Science, Gauhati University, Assam, India. His research interests include representation of web documents using graphs, graph based clustering and graph database.