Hybrid Distributed Techniques for Lower-dimensional Representation of Documents in the Scientific Literature

by

Bahareh Kazemi

Bachelor of Science, University of Mazandaran, 2011

A thesis

presented to Ryerson University

in partial fulfillment of the

requirements for the degree of

Master of Science

in the program of

Computer Science

Toronto, Ontario, Canada, 2019

AUTHOR'S DECLARATION FOR ELECTRONIC SUBMISSION OF A THESIS

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I authorize Ryerson University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize Ryerson University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

I understand that my dissertation may be made electronically available to the public.

Hybrid Distributed Techniques for Lower-dimensional Representation of Documents in the Scientific Literature

Master of Science 2019

Bahareh Kazemi

Computer Science

Ryerson University

Abstract

Surfing data mining techniques for representing data sources have specifically attracted much attention among researchers. Given the curse of dimensionality in representing text using the traditional Bag-of-words models, lower-dimensional representation of text has been an important line of research due to its impact on many prediction, and recommendation tasks.

This thesis studies two main different viewpoints in text representation using content and citation information and then, different existing approaches along with their advantages, limitations and drawbacks are reviewed. A novel hybrid distributed technique for text representation is proposed where the textual content of documents is projected into a vector representation using an artificial neural network .

To test the performance of the new proposed technique, the well known link-prediction problem is selected to serve as a benchmark. A comparison is performed with other

common techniques by predicting the existence of citation links between tuple of papers in a large citation graph.

Acknowledgements

I first would thank my supervisor Professor Abdolreza Abhari for allowing me to work under his supervision during my master's.

I also want to thank the members of my committee Professor Eric Harley, Dr Bahram Zahir and Professor Cherie Ding for generously offering their time to review this thesis.

I have been also fortunate to collaborate with my colleagues at DSMP lab (dsmp.ryerson.ca) of Ryerson university. I am specially indebted to the members of the lab for forming a friendly and vibrant atmosphere in which I was able to conduct a significant part of my research.

My special thank goes to my parents for encouraging me constantly to pursue my academic career. And last but not the least, I should express my deepest gratitude to my husband, Ali, for all love, support and motivation. I would certainly not be able to do this without his emotional support.

Dedication

To my two little kids Noyan and Mani whom I missed frequently while doing this research

Table of Contents

List of Tables

List of Figures

List of Acronyms

AI	**Artificial Intelligence**
ANN	**Artificial Neural Network**
AUC	**Area Under Curve**
AWS	**Amazon Web Service**
DFS	**Depth-first Search**
BFS	**Breadth-first Search**
HMM	**Hidden Markov Models**
HTTP	**Hypertext Transfer Protocol**
IEEE	**Institute of Electrical and Electronics Engineers**
IMDB	**Internet Movie Database**
LDA	**Latent Dirichlet Allocation**
LINE	**Large Scale Information Network Embedding**
LSA	**Latent Semantic Analysis**
LSTM	**Long Short Term Memory**
MLP	**Multi Layer Perceptron**
MS	**Microsoft**
NLP	**Natural Language Processing**
NLTK	**Natural Language Toolkit**
TFIDF	**Term Frequency-Inverse Document Frequency**

Chapter 1

Introduction

This chapter is dedicated to review the main motivations and challenges of the research conducted in this work. The motivation is discussed in Section 1.1. The general problem statement is given in Section 1.2 where common challenges and objectives of the formulation are provided in Section 1.3. A brief overview of the proposed methodology, candidate data-set for evaluation and implementation details is given in Section 1.4. Section 1.5 explicitly goes through the contribution of this thesis and finally these parts are followed by section 1.6 which demonstrate the thesis organization.

1.1 Motivation

Nowadays, with the vast accessibility of scientific literature and broad and diverse areas of science explored by researchers, the necessity for deeper knowledge extraction from the articles is crucial. Beside traditional Information Retrieval (IR) and machine learning techniques developed for document clustering [1, 2]and classification [3, 4, 5], disambiguation processes [6, 7], concept extraction [8] and recommendation engines [9, 10, 11], with the emergence of artificial neural networks and their contribution to Natural Language Processing (NLP) [12], there has been recently a track of research on developing new architectures for a universal paper representation that could be applied to any of above-mentioned areas.

Traditionally, document modeling aims to locally represent each document regardless of its contextual interaction with other documents. While in local representation

techniques each unit is represented considering only its own appearance or frequency through out the text of the documents, on the other hand, with the appearance of artificial neural networks and data-driven techniques such as deep learning methods, families of documents representation that consider semantic relations, which are also referred as distributed representation techniques, have become an important line of research in Natural Language Processing (NLP). While a significant part of research has been based on developing algorithms for representing documents through either the embedded lexical relationships in the text or contextual relations between documents as part of an arbitrary citation network, less effort has been spent on combining both aforementioned viewpoints and building a new distributed approach that leverages both the content and citation information.

The main motivation of this thesis is to conduct research on the application of modern technologies such as deep learning in leveraging both the documents textual content and relationship information through citation graphs to build a universal and efficient architecture for document representation. In addition, this thesis makes a detailed analysis and comparison over the existing families of techniques for document modeling and shows the efficacy of the proposed approach on a real data-set of scientific articles in the field of computer science.

1.2 Problem Statement

One of the most crucial stages in building an architecture to analyze textual documents is to propose a proper feature vector representation of text. This usually refers to a procedure that enables us to represent a document mathematically(feature vector) in a way that it can be used by many recommendation or prediction engines. In this case, any document may represent many types of information and a feature vector representation can project an arbitrary document into a form that all the information is measurable. The main questions arising during this process are as follows:

- How can one specify a mapping function f that can project an arbitrary document d to a feature vector representation e?
- How compact can the representation e be while being able to retrieve maximum amount of information embedded in the original document d?

- Is it sufficient to find the mapping function f locally and based on only the content of the specified document, or can we form the document representation based on a distributed relationship of documents across a given network of citation relationships?

This thesis tends to formulate a novel mapping mechanism f that projects a bag of words-based local representation [96] of the text to a compact and distributed vector representation. Therefore, the main problem discuss in this work is to design a framework to learn a mapping function f that produces vector representations e of arbitrary textual documents d. Figure 1.1 shows a generic block-diagram of the technique used in this thesis for document representation where it shows how mapping function f uses the citation graph and content to generate vector representation of a certain document d.

1.3 Objective and Challenges

In this thesis we aim to study a family of existing techniques for modeling and representing documents that are normally defined as a collection of coherent textual information. Essentially, these techniques are employed to be able to grasp any informative aspect of a document that can be captured or comprehend by a human user.The common drawbacks and limitations to existing approaches have motivated us to propose our novel hybrid method with an effort to leverage those downsides and present a more solid representation vector. The existing techniques usually study document representation by either analyzing the textual content of each document or considering the document in a larger relation-graph that shows the documents interaction with other documents through context-specific citations. Any of the above-mentioned strategies has its own advantages and disadvantages and, therefore, may be used in a certain task.

1.3.1 Content-based Representation

These types of representations mainly focus on the textual content of the documents and aim to grasp any embedded potential knowledge in the text. There are several approaches categorized as content-based methods, all of them doing the knowledge extraction based on the documents text while approaching the task differently. The traditional TF-IDF

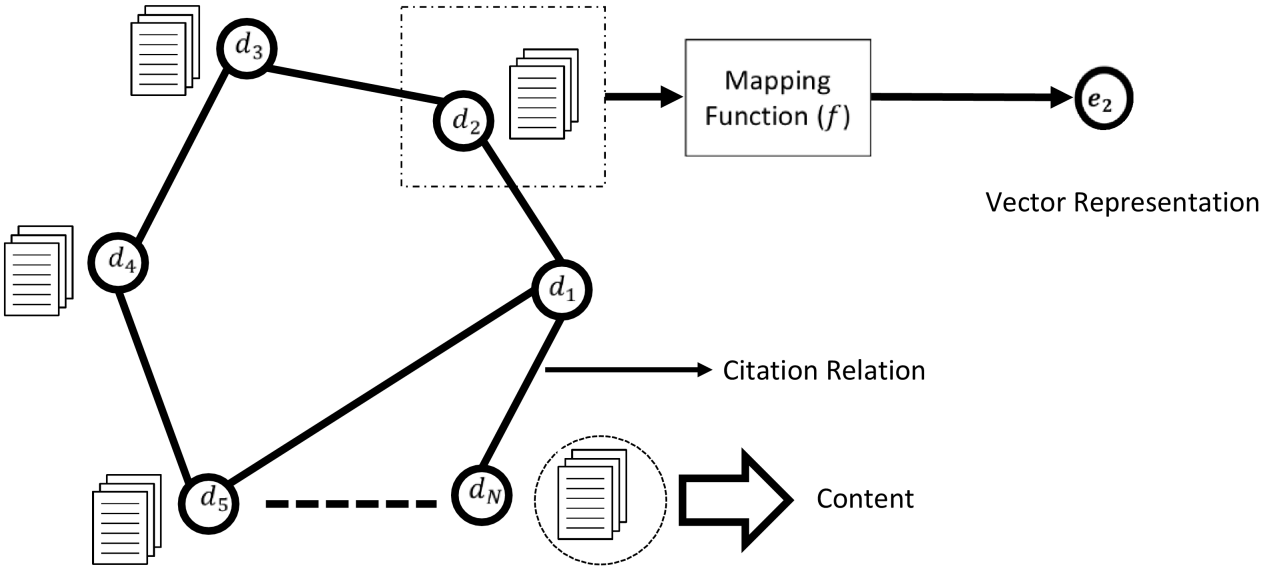


Figure 1.1: A generic diagram of the distributed hybrid technique for generating document representation using both the content and citation information.

method is the most widely used technique for document representation where each document is projected to a highly dimensional vector of N by 1 with N being the size of the dictionary of words. Although ease of use and no need for any training interface are taken as the main benefits of TF-IDF method, the approach fails to model some certain characteristics of the text such as semantic content, and is not cost effective.

Other category of content-based techniques concentrate on representing the text considering the semantic and grammatical relationships between text units embedded in the text. Methods falling in this category usually need to proceed with an intermediate training stage to project each textual unit (eg. words, sentences, etc) to a lower-dimensional vector [98]. Techniques such as Word2Vec, Doc2Vec, topic modeling and Latent Drichtlet Allocation (LDA), and sequence modeling using Artificial Neural Networks (ANN) usually use a training corpus to learn the representation of the text in the corpus.

While the above-mentioned family of techniques outperform the traditional TF-IDF by embedding semantic context of text in their vector representation [95], they do not still consider non-textual knowledge and, hence, lose some useful data that is normally found in term of document-to-document relationship. As an example, consider the world of scientific papers in two different fields, one on soccer and the other regarding leg muscle diseases. The papers seem semantically unrelated prima facia, while in reality, assume that it is discovered there is a chance for those playing soccer to increase the chance of being diagnosed with a specific form of muscle disease in elderly days. Expectedly, such a relationship can be interpreted from a citation graph where a paper in the field of soccer cites another article discussing muscle diseases. Although the relation between above-mentioned articles cannot be extracted through their content-based semantic representation, a citation graph can provide strong relation signal between the documents. Therefore, while content-based techniques provide powerful representation methodologies, they cannot still include some valuable relation information usually embedded in a citation graph.

1.3.2 Citation-based Representation

Citation based representation techniques emphasize on relations across documents that are usually provided through a large citation graph. The idea behind this approach is to represent each document based on its interaction with its neighbor documents within the

graph. Note that neighbors in this context can be referred to those documents that have citation relations with each other. In general, considering any arbitrary document, majority of its citations are belong to those documents focusing on the same area of research as the citing document, which can be inferred that these papers are conceptually related. Therefore, citation-based representation approaches aim to mitigate the drawback in content-based techniques due to the lack of considering citation relations in creating the vector representation. The research on the citation-based techniques has been mainly focused on mapping a large and sparse citation graph to a lower dimensional representation. matrix factorization techniques are popular as they do not normally require any training data and can be applied to an arbitrary graph to produce vector representations [97]. With emergence of Deep Learning (DL) methods, recently, techniques such as deep Collaborative Filtering (CF) [91], DeepWalk [80], and node2vec [27] have been proposed to create node representation within a graph through learning the vector projection by considering the node citations within the graph. For example, node2vec admits each node as an N by 1 vector with the only non-zero entry corresponding to the index of the node. Then, an ANN learns to produce neighbor nodes in the graph by projecting the input to a lower dimensional space that is called node embedding. Unlike the traditional matrix factorization techniques, DL-based methods require a vast amount of training data.

Although the citation-based techniques show effective in tasks with existing citation information, they still suffer from some drawbacks. First they are affected by the cold-start phenomenon that is related to their weakness in modeling the newly added nodes to the graph, the ones with little citation information. Moreover, using only citation information without considering content can be sub-optimal, especially, considering no information about the type of citation is available (whether it is a positive citation or a negative one). As a result, a more comprehensive approach is required that can leverage benefits of both two above-mentioned methods and is able to represent as much informative aspects of each document as possible.

1.3.3 Hybrid Techniques

Hybrid techniques are, generally, referred to approaches that can combine different methods, such as content and citation based techniques, to bring in multiple advantages all together and produce more solid results compared to citation/content only techniques.

In recent years, there has been an increasing favor among researchers toward hybrid approaches due to their potential capabilities in performing complicated tasks in data analysis and machine learning problems. For instance, there is one family of hybrid techniques that aims to form document representation for similarity calculation in recommendation system applications. This group of hybrid approaches considers the citation relations of the paper within a citation graph, alongside their context to measure the similarity of the documents. Therefore, unlike the conventional citation-only based techniques, the relation between each two nodes is weighted based on the content of each of the nodes. Then, the matrix factorization can be used to map each high-dimensional $N \times 1$ document vector into a lower-dimension of H with $H \leq\leq N$. Here for any target document, the n-th entry of the high-dimensional representation vector corresponds to a similarity score between the target document and the n-th document in the graph, where the score is also calculated based on the existence of citation relations and similarity of their texts.

Although the above-mentioned approach and all the similar hybrid techniques establish a significant knowledge coverage while using both the text and citation interactions, their implementation can be challenging as they can significantly require feature engineering to calculate the relation weights. This means the efficiency of the method is affected by the way the similarity score is formulated. In addition, it is not known how the text and citation based features can be combined to form the final node similarity.

Beside the above issue, another drawback for these families of hybrid approaches is the curse of dimensionality. As each document is required to admit a vector in the size of the number of nodes in the citation graph, these techniques do not normally result in a scalable solution. On the other hand, another family of hybrid techniques aim to project each document into a lower dimensional vector (feature vector), which is the main contribution of this thesis. The idea presented here is to use both the content of the document and citation information where a solution using a neural network model is employed for training purposes. The neural network model here admits each target document in the citation graph as the input and, during the training phase, it learns how to project each document to a lower-dimensional vector representation that can predict the word distribution in the surrounding (neighbor) documents in the output of the network. The proposed framework can form a distributed solution to the document representation problem where both the citation and textual content of the documents are used in the vector representation.

1.4 Methodology and Implementations

In order to evaluate the proposed technique in this thesis, we need to discuss the methodological steps and implementation details. In this section, we first start with an overview of the architecture and, then, discuss data-sources, preparation phases, evaluation and software implementation details, of course in a general manner.

1.4.1 Architecture and Training

The proposed architecture in this thesis utilizes an ANN to produce vector representation of documents. The ANN is trained in a way that the final trained model can admit the document's text and map it to a lower-dimensional vector. In this case, the ANN is represented as a feed-forward MLP network that maps input text representation to a document vector. Figure 1.2 shows the block-diagram of the pipeline for document vector representation.

To tune the weights of the above-mentioned MLP network, training tuple is generated by considering each arbitrary document and its neighbors. In this context, a neighbor refers to those documents that have established citation relations among each other and are ,hypothetically, conceptually related. As Figure 1.2 shows, the proposed structure consists of four different stages as follows:

1. *Input layer*: this layer admits the textual content of the document d_0 as the input and, then, projects the text to a bag of words vector that is used as the vector representation sent to the embedding layer.

2. *Embedding layer*: the bag of words vector is then applied to the first layer with parameters θ_1 which needs to be tuned during the training. The document's embedding $\mathbf{e}_0$ is then generated as the output of the embedding layer.

3. *Softmax layer*: the document embedding vector is then applied to the output layer that projects the embedding vector to an output bag of words vector. The output of this layer is a bag of words vector with the same size of the dictionary of the words where the i-th entry of the vector corresponds to the assigned weight to the i-th word of the dictionary.

4. *Training phase*:the cost function is created by considering the real bag of words vector of all neighbor documents $\{d_1, \ldots, d_N\}$ and the produced bag of words vector in the out put layer. Parameters of the first layer (θ_1) and the output layer (θ_2) are tuned such that the inner layer embedding vector will result in an output bag of words vector with higher probabilities assigned to words appearing in the neighbor documents.

Once the training phase is finalized, the embedding layer is now capable of producing a lower- dimensional vector of the input document. This means the embedding vector produced in the inner layer is representing each input article based on its neighbor (similar) documents. Since each document is going through the model with its content-based representation and outputs the result based on the citation relation, it benefits from both aspects and is expected to drive more solid performance than non-hybrid methods.

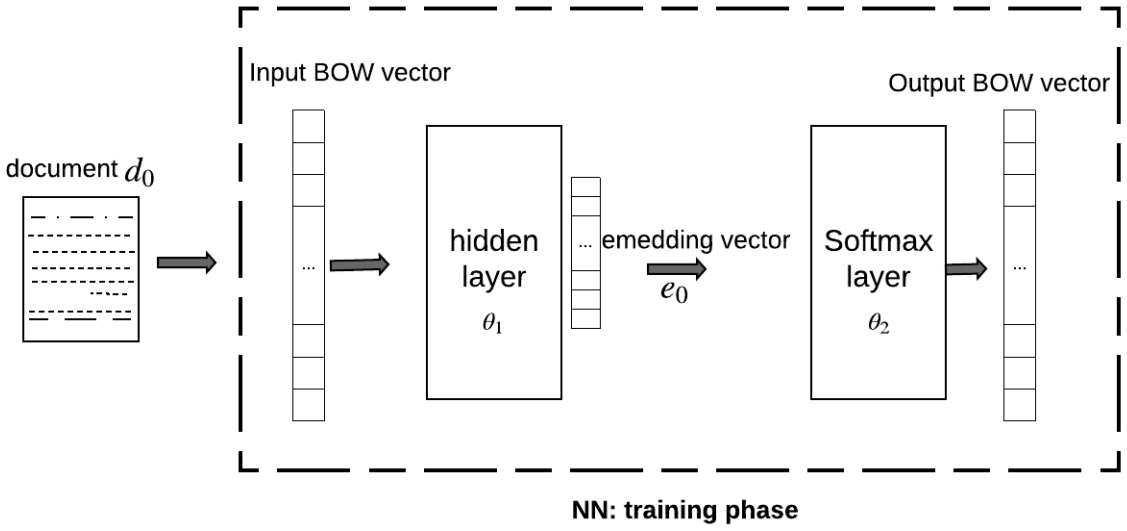
1.4.2 Data Source

For the purpose of training and performance evaluation, articles from a list of well-known journals and conferences in the field of computer science over the last 10 years are chosen through the Microsoft Academy (MS) knowledge graph [39]. The MS graph also provides both the abstract and the citation relation information for all the articles in the pool. For the purpose of the experiments in this thesis, only those citation records where both citing and cited articles exist in the paper-pool are considered.

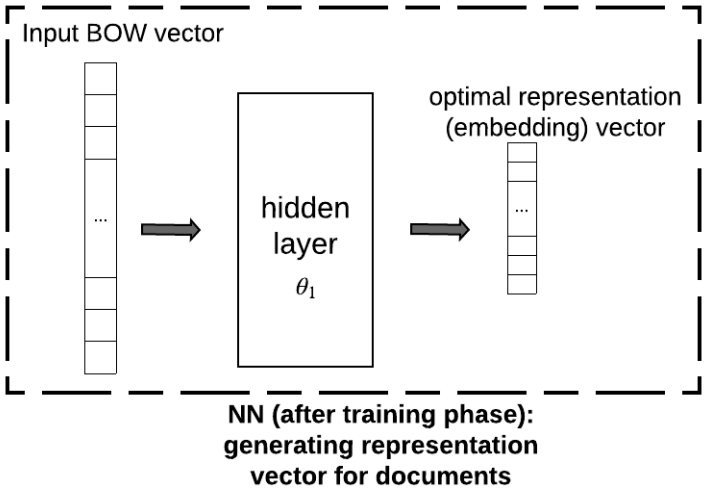
1.4.3 Data Preparation

Once the paper pool is produced and citation relations are extracted, a series of different data preparation stages is applied to the raw-text:

- *Lemmatization and stemming*: To preserve only the basic form of the words and increase the coverage of words across articles, each word goes through a series of lexical processing steps such as stemming and lemmatizition. As a result, all the redundant forms of a certain word are eliminated and, then, grouped to a unique term that can be added to the dictionary.



(a)



(b)

Figure 1.2: A block-diagram of the pipeline used in this thesis for learning the vector representation of documents from the textual and citation information. Both (θ_1) and (θ_2) corresponds to a set of parameters for their assigned layer.

- *Dictionary generation*: a dictionary containing all unique words in the pool of candidates is created. To reduce the computational cost, only those words with a certain number of appearances in the articles are included in the dictionary.

1.4.4 Performance Evaluation

The efficacy of the generated document vectors using any of the presented techniques in this thesis is evaluated against the well-known link-prediction problem where the goal is to predict the existence of a link (edge) between any two nodes (papers) in the citation graph. For this prediction problem, the direction of the edge is ignored and the existence of a citation is considered as a relation between each two certain papers.

The link-prediction problem can be now formulated as a binary classification scenario. The logistic regression classifier is first applied using the feature vectors obtained by each specific model that produces vector representation of documents. AUC is also used as the classification metric to evaluate the performance of the link-prediction problem.

1.4.5 Implementation

A pipeline is formed to produce representation vectors and run link-prediction tasks. Each stage of the pipeline can be explained as follows:

- *Document extraction*: all the textual data and citation relationships are extracted from MS academy pipeline through Python HTTP requests. We obtained the list of top-50 journals and conferences in the field of computer science and extracted published papers by any of the venues over the last 10 years. Raw text of each paper as well as citation data are both extracted from MS academy pipeline.

- *Pre-processing and data preparation*: to run common pre-processing steps such as data-cleaning, lemmatization and stemming, Python's NLTK [40] package has been used.

- *Model training*: the content-based Node2Vec model is implemented in tensorflow [41]. We also use the NLTK package to run experiments for topic-modeling and LDA.

- *Link-prediction and performance evaluation*: the link-prediction problem has been modeled as a binary classification scenario and is dealt with using an logistic regression classifier. To implement the logistic regression classifier, we use the built-in function in Python's ScikitLearn [42]. For performance evaluation, the AUC has been also calculated.

Given the computational complexity of the ANN training and requirement for running cross-validation tasks, the whole training and cross-validation run on an M4 AWS machine with 16 cores and 256Gb of RAM. The above-mentioned pipeline design enables running multiple parallel jobs on the machine to evaluate the suitability of each set of parameters.

1.5 contributions

The main contribution of this thesis is listed as follows:

- The paper proposes a novel hybrid technique with extending the original idea in [27] to include textual data along with the citation information in the training phase to lead building a more efficient feature vector.

- The technique has addressed the drawbacks that the original Node2vec was dealing with such as cold start problem and dynamically sized graph.

- A rigorous analysis is conducted on the advantages of the newly proposed structure compared to the traditional content and citation based techniques. The efficacy of the novel hybrid representation technique is specifically, tested over an edge prediction problem compared to the existing traditional approaches that are implemented. in this work.

1.6 Thesis Organization

The rest of this thesis is organized as follows.Chapter 2 presents background information and related works where content based representation using TF-IDF and word embedding and citation based representation using node2vec and DeepWalk are reviewed. The

main contribution of this work is discussed in chapter 3 where the methodology used to implement content based node2vec is presented.Simulation results are also provided in section 4 where the edge prediction problem is presented as a comparison benchmark. Section 5 also conclude the thesis and suggests the future work.

Chapter 2

Literature Review

The accessibility to human knowledge has become much more facilitated with the emergence of internet technology. Nowadays, people enjoy wide access to a variety of information sources such as news, science, literature, movies and public services through different mediums such as text, images, videos and voices. Given the variety of information sources and diversity of the services required by people, the modern technology is now moving towards proposing novel ways to process information media and facilitate their usage and interpretation for ordinary citizens.

Textual data have become a unique source of information, especially, with the current growing pace in Information Technology (IT). Unlike many other sources such as videos or voices whose ages are not more than a century, human being has traditionally recorded its knowledge through scripts or hand-writing. Nowadays, a significant amount of knowledge and information can be found in old manuscripts whose textual data has been widely used in the modern scientific community. Many scientific publishing groups are now working on digitizing the old corpora and form them as individual documents as part of a large scientific graph. Most of old written articles in medicine and biology are now stored and indexed by PubMed [38]. ThomsonReuters has also built Westlaw [43] as a knowledge-base for lawyers and attorneys where millions of case laws written by judges and attorneys have been digitized as law documents. Many other publishing groups such as Elsevier [45], IEEE [44] and Nature [46] have all dedicated their resources in building platforms that can efficiently store and index textual data. With such a fast pace of access to text information, the necessity for inventing novel techniques to analyze the

text and mine the desirable information has become more urgent.

This section reviews the state-or-art technologies in document modeling and, especially, different view-points in building lower-dimensional representation of text. First, background and literature review is provided and, then, related work that motivated the main contribution of this thesis is reviewed.

2.1 Background Information

To analyze any information source, such as scientific papers, the first crucial step is to build a numerical representation vector, generally called feature vector, for each document. Unlike other sources of information such as videos and images, text representation faces more challenges given the ambiguity in the representation stage (word, sentence, paragraph, etc) and representation context (local or distributed) [47]. Considering words as the smallest meaningful unit of the text, the traditional text representation has been based on forming models to quantify words. Higher-level representations such as sentence or paragraph level can be then obtained by aggregation over individual words. With a significant progress in research on Natural Language Processing (NLP) and Artificial Intelligence (AI), and with wider access to the text data, researchers have proposed more advanced ways to model larger level of text such as sentences [21, 22, 48] and paragraphs [19, 22, 25, 49] given their relation and existence within a given certain text corpus.

Beside the representation stage, there has been recently a significant research on distributed representation of text. Most of the traditional techniques model each text unit such as word as a one-hot vector of size of a human-formed dictionary where the m-th entry of the vector corresponds to the existence of the word in the dictionary. Indeed, these families of methods do not consider the interaction between each word and other words within the text in forming the representation. In this case, as the representation of each word is only considered by its local index within a formed dictionary of words, such techniques are also called local [47]. Local models suffer from a number of drawbacks such as curse of dimensionality and lack of universal representation context. As local methods usually form a very large vector of word indexes whose size is equal to the size of the dictionary, they normally lead to propose high dimensional vectors and, therefore, face challenges such as huge disk/memory space requirement. Also, another drawback

is in classification problems, where normally there are not many labeled data available for training the classifier. As each word context is formed independent of its relation to other words, the ultimate word representations normally form a disjoint space of word vectors. In such cases, the classifier is dealing with large number of parameters to learn (same size as the input vector), while it does not have as large labeled training data to be able to learn properly.

Given the existence of large collections of textual data, there has been efforts to use the relations among textual units (by unit we mean any chunk of text such as words, sentences, paragraphs, etc) to obtain a vector representation of the text that also considers both the lexical and semantic textual relationship. Unlike the local method, this family of techniques computes the text representation based on the relationship among text units and, therefore, is called distributed. Note that depending on the way the distributed representation is tackled, different types of relationships can be considered. For example, surrounding words within a sentence or paragraph can be considered semantically similar and the vector representation can be defined in a way that the surrounding words are projected to the same cluster of vectors. On the other hand, other types of relationships such as co-occurrence of words within the same document can be considered in building a distributed representation.

2.1.1 Content Based Representation

A large body of research on document representation has been focused on leveraging the textual content of each document for building a representation. In the following, both the local and distributed view-points are reviewed in the context of content-based representation.

Local Content Based Representation

Local representation of text using bag of words has been traditionally used in document modeling [50]. The bag of words model projects each document to an $M \times 1$ vector of integers where M corresponds to the size of the dictionary of words and each word is also modeled by an $M \times 1$ vector of all zero values with the only non-zero entry denoting that word's index in the dictionary. Each entry of the bag of words model then equals to either the frequency or existence of the word in the document. The bag of words

model has been vastly used in many document retrieval/classification tasks [51],[52],[53], especially, in supervised learning problems with labelled documents. The bag of words model has been also used as the input unit for many of distributed learning procedures where the document is fed to the model through a bag of words vector.

The bag of words model can be then extended to the well-known Term-Frequency Inverse-Document-Frequency (TF-IDF) [12] model by considering the frequency of each word across different documents through an IDF measure [12]. The original TF-IDF aims to project each document to an $M \times 1$ vector where M corresponds to the size of the dictionary of words and each entry is a multiplication of TF and IDF values. The TF-IDF text representation has been used as the basic vector representation in many applications such as web-search [54], document classification [55], content-based document recommendation [56], text similarity calculation [57], and sequence learning [58]. TF-IDF is implemented as one of the common traditional approaches in this work and below a brief explanation on how it has been modeled is provided.

Definition 2.1. *Considering d_i and $\{\bar{w}_1, \ldots, \bar{w}_M\}$ as the i-th document and set of all the words in the dictionary, respectively, define $\boldsymbol{t}_i$ as an $M \times 1$ vector of integers where t_i^m represents the frequency of w_m (the m-th word) in the i-th document.*

Definition 2.2. *Assuming the set of all the documents $\{d_1, \ldots, d_N\}$, define idf_m as the IDF of the m-th word in the dictionary as*

$$idf_m = \log\left(\frac{N}{n_{\bar{w}_m}}\right) \tag{2.1}$$

with $n_{\bar{w}_m}$ being the number of documents that contain $\bar{w}_m$.

The TF-IDF vector $\mathbf{v}_m$ can be now defined for each document as an $M \times 1$ vector where the m-th entry is given by

$$\mathbf{v}_i^m = \mathbf{t}_i^m \times idf_m \tag{2.2}$$

A summary of TF-IDF algorithm for document representation can be found in Algorithm 1.

While none of bag of words and TF-IDF models require training procedures to create their vector representations, they suffer from a number of drawbacks. First, both models

Algorithm 1 TF-IDF Algorithm for document Representation

0: **Inputs:** Set of documents $\{d_1, \ldots, d_N\}$ and associated raw-text
0: **Parameters:** Size of dictionary (M) and IDF threshold (τ)
0: **Outputs:** An $M \times 1$ TF-IDF representation for each of the documents in the set
0: **Dictionary Generation:**

- **Form set of all the words from N documents as $\{w_1, \ldots, w_I\}$ with I being total number of distinct words**
- **Apply pre-processing steps lemmatization and stemming to form a sub-set of J distinct words $\{\bar{w}_1, \ldots, \bar{w}_J\}$**
- **Calculate *idf* of each word and remove all the words with an IDF below τ**
- **Choose M words from the filtered set and form $\{\bar{w}_1, \ldots, \bar{w}_M\}$ as the dictionary of words $\mathcal{D}$**

0: **TF-IDF Calculation:**
for i in N **do**
Extract all distinct words, apply lemmatization and stemming, and form the set $\{\bar{w}_1, \ldots, \bar{w}_{N_i}\}$
Initialize $\mathbf{v}_i$ by an $M \times 1$ vector of zeros $\mathbf{0}$
for j in N_i **do**
if $\bar{w}_j$ in D **then**
$v_i^{m_j} += idf_{m_j}$ where m_j denotes the index of the j-th word in the dictionary
end if
end for
end for=0

provide a high dimensional vector with a size being equal to the size of the dictionary and given the size of the document, they can potentially have large non-zero entries. This will normally require a significant amount of disk/memory storage, especially, in tasks with a large number of documents. In addition, none of bag of words and TF-IDF methods consider the semantic meaning [47] of words in their vector formation as the word-modeling is based on the local on-hot representation. Therefore, both methods are unable in measuring the semantic relatedness of two given documents and may not be efficient for many tasks involving semantic relatedness such as recommendations and text similarity(in cases that the exact same mention of words are not used). Finally, bag of words and TF-IDF cannot model the order of apperance of words in the sequence of words in the text. Therefore, unlike many language modeling frameworks, such as Recurrent Networks [14], which take the above sequential relationship into consideration, both bag of words and TF-IDF do not oversee the sequence of relations among words in the text.

Distributed Representation

Distributed learning and representation of text has been an active area of research [47],[59],[60]. Unlike the local representation techniques in which each unit of text is modeled by its index across a pre-defined dictionary, distributed techniques project each unit to an arbitrary-sized vector by considering its relationship with other parts of the text. It has been shown that such a modeling not only does consider semantic meaning of the text but also provides more compressed vector representation that can save both time and hardware [47]. Given different interpretations of textual relationships across existing corpora, several approaches have been proposed for distributed representation of text.

One of the earliest efforts on the distributed representation of text has been based on relating words based on the their co-occurrences in the text documents. Topic modeling [15] has been then proposed as a general approach to cluster similar words and assign them to a more abstract index called topic. As one of the most well-known topic modeling approaches, Latent Semantic Analysis (LSA) [61] first creates a term-to-document matrix where each entry represents the occurrence of the token within a certain document. In this case, other local representation techniques such as TF-IDF can be also used to determine the weight of each term within a document. As the main occurrence matrix is

normally a very high dimensional matrix, LSA proposes efficient ways such as Singular Value Decomposition (SVD) to project the original high-dimensional matrix to a lower-dimensional form where each term can be represented by a much more compact vector. LSA has been then used in many NLP applications such as text summarization [62], document classification [64], and relationship discovery [63], etc.

Document representation using LDA

While LSA is intrinsically a non-probabilistic approach and may also suffer from the scalability due to the inherent computational complexity of SVD, the techniques such as Latent Dirichlet Allocation (LDA) [15] have been also proposed to handle topic-modeling in a more probabilistic way. The LDA creates a generative process that can produce documents from some pre-defined distribution of words called topics. Once the generative model is formed, LDA uses text of documents as the training data and estimate hidden variables such as topics. Then, each document can be represented as a low-dimensional vector where each entry corresponds to the weight of the topic assigned to the underlying document. The theory of topic modeling proposes that each document consists of a finite set of topics that are each characterized as a distribution over the words. Considering a topic as a distribution over the words, the well-known Latent Dirichlet Allocation (LDA) [15] creates a generative process that assigns each single document to a set of words drawn from the mathematical distribution of topics over the words. The LDA model then extracts a topical weight vector for each document in the corpus that could be used as a vector representation of the document in the scientific network. In the following we dig in to some details on how a LDA model is built:

Definition 2.3. *Given the word dictionary of size M and N as the total number of documents in the network (aka corpus), the following topical notations are defined:*

- *An $M \times K$ matrix ϕ with ϕ_{mk} being the probability of assigning the m-th word in the dictionary to the k-th topic.*
- *A $K \times 1$ vector θ^n with θ^n_k being the probability of assigning the k-th topic to the n-th document.*
- *A Dirichlet prior on topic-to-document assignment $Dir(\alpha)$ with α being a $K \times 1$*

vector that determines the prior information on the importance of each certain topic within a document.

- *A Dirichlet prior on topic-to-word assignment* $Dir(\beta)$ *with* β *being a* $M \times K$ *matrix that determines the prior information on the relevance of each certain word to a topic.*

LDA is a probabilistic generative model that produces natural-language documents from the dictionary of words through the following steps:

- *Parameter*: number of words appearing in the n-th document (U_n)
- *Topic assignment*: draw a topic-to-document distribution θ^n randomly from $Dir(\alpha)$
- *Word assignment*: draw a topic-to-word distribution ϕ randomly from $Dir(\beta)$
- While $u < U_n$ do:
  - Draw a topic z_u^n randomly from the Multinomial distribution θ^n ($z_u^n \sim$ Multinomial(θ^n))
  - Draw a word w_u from the distribution $\phi_{:z_u^n}$ with $\phi_{:z_u^n}$ representing the z_u^n-th column of the probability matrix ϕ

The above steps can be repeated for all N documents in the corpus.

In a real scenario, the documents and assigned textual information are all available. The goal is to estimate the following hidden variables and parameters:

- θ^n: the distribution of topics over the n-th document
- z_u^n: the topic assigned to the u-th word in the n-th document
- ϕ: matrix of word-to-topic assignment
- α: parameters of the Dirichlet prior on the topic distribution
- β: parameters of the Dirichlet prior on the topic-to-word distribution

Given N documents in the corpus, K topics, and assuming U words assigned by average to each document, there are $N \times K + N \times K \times U + M \times K$ hidden variables to estimate. The hidden-state and parameter estimation in LDA can be done using different

computationally efficient techniques such as variational Bayes [66], and Gibbs sampling [34]. Algorithm 2 shows steps to generate a $K \times 1$ vector representation of documents using LDA algorithm. The final lower-dimensional vector can be used to represent the document in the corpus of scientific network. Beside further developments of the original LDA to other cases such as supervised LDA [65] and online LDA [66], the technique has been applied to a large number of applications such as document clustering [67] and information retrieval [15]. As topic-based models cluster terms based on their co-occurrence

Algorithm 2 Topical Weight Extraction for document Representation

0: **Inputs:** Set of documents $\{d_1, \ldots, d_N\}$ and associated raw-text
0: **Parameters:** Size of dictionary (M), and number of topics (K)
0: **Outputs:** A $K \times 1$ topical weight representation for the document
0: **Dictionary Generation: Form a dictionary of M words following the procedure given in Algorithm 1**
0: **LDA training:** Train an LDA model with K topics against all the N documents in the training pool
0: **Topical weight extraction:**
for i in N **do**
Gather set of all the words appearing in the i-th document as $\{w_1, \ldots, w_U\}$
Using the trained model, find all word-to-topic assignments $z_u \in \{1, \ldots, K\}$
Calculate the weight of each topic by

$$p(\theta_k^i | w_{1:U}) \sim \frac{\sum_u I^k(z_u)}{U} \tag{2.3}$$

with $I^k(z_u) = 1$ if $z_u = k$ and 0 otherwise.
Output a $K \times 1$ vector of topic weights $[p(\theta_k^i | w_{1:U})]_{k \in \{1,\ldots,K\}}$ as the lower-dimensional representation of the i-th document
end for=0

in a document, they may still miss modeling a proper semantic relationship between the terms. That two terms co-occur across a large body of text does not necessarily mean that they are strongly related. Therefore, recent research in distributed learning has been focused on proposing ways to better model the relationships across the text. Language models [68] usually form a probabilistic model that can predict a word based on its neighbors. These models normally consider the underlying Grammar in the text and can be used in many other models such as Hidden Markov Models (HMM). Given the embedded hierarchy in the text, these models may consider simple unigram to more

complicated hierarchies such as bi/n-grams [12]. As these models usually calculate the probabilities based on the occurrence of word sequences in the corpora, they need a large amount of training data to be able to have a good estimate of word probabilities.

With the recent advances in the field of ANN and DL [14], the attempts for building data-driven representations of text have become more significant. The seminal Word2vec model was first proposed in [18] as a systematic way of generating lower-dimensional vector representations of words. The idea has been also extended to phrases and paragraphs in [19] where the well-known Doc2Vec model creates paragraph vectors based on their relationship with word vectors. Unlike the topic-based models, both Word2Vec and Doc2Vec techniques represent a word based on its relationship with its neighbor words in a training corpus. Therefore, words that are located in a closer proximity will admit vector representations that attain smaller distance. Measures such as Cosine-similarity have been then used to calculate word similarities based on the generated word representations and have been applied to many semantic similarity tasks [69],[70]. Other variations of the Word2Vec model have been also developed in the literature to consider specific types of semantic relatedness such as synonym/antonym [71]. While Word2Vec based models have shown a great success in representing semantic relatedness between words, their extension to larger bodies of text such as documents is still an area of research.

Document Representation Using Word Embedding

The main goal in generating word embedding is to address the drawbacks in the traditional TF-IDF representation that ignores the semantic relationship between words within the textual data. The well-known word2vec [18] model is formed based on the assumption that neighbor words within the text are semantically correlated.

Definition 2.4. *For an n-th paper p_n in the scientific network, assume $\{w_1, w_2, \ldots, w_M\}$ as an ordered set of words where $\{w_{m-1}, w_m, w_{m+1}\}$ being considered as spatially neighbor words in the document.*

The main idea in building a word embedding model is to follow one of the logics below:

- *Skip-Gram (SKG)*: considering a window of surrounding words $\{w_{m-L}, \ldots, w_{m-1}, w_{m+1}, \ldots, w_{m+L}\}$ for w_m, the goal is to create a model that predicts the surrounding words based on the given target word w_m.

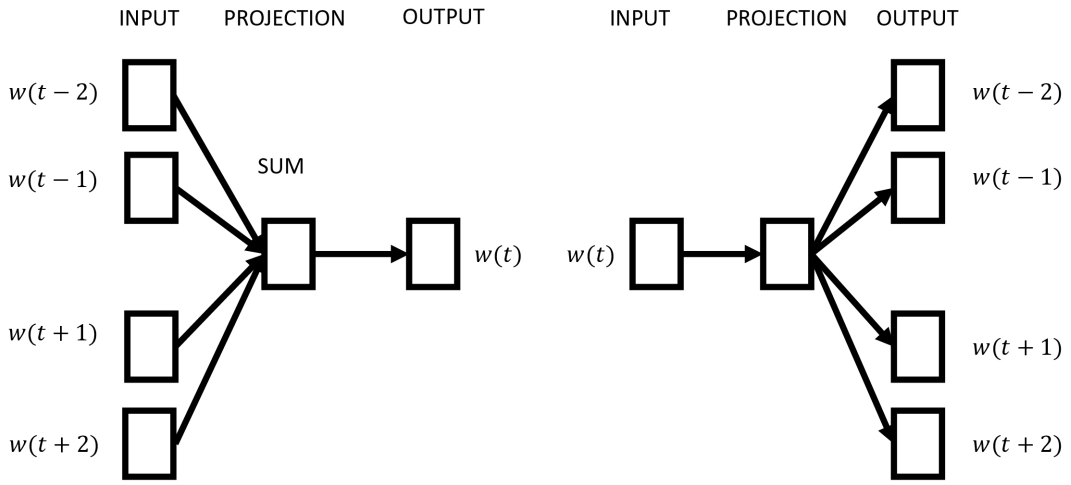


Figure 2.1: Wordtovec models diagram for both skip-gram and continuous bag-of-words [18].

- *Continuous Bag-of-Words (CBOW)*: considering a window of surrounding words $\{w_{m-L}, \ldots, w_{m-1}, w_{m+1}, \ldots, w_{m+L}\}$ for w_m, the goal is to create a model that predicts the target word w_m based on the given surrounding words.

In both structures, the network expects a tuple $\big((w_{m-L}, \ldots, w_{m-1}, w_{m+1}, \ldots, w_{m+L}), w_m\big)$ as the training instance. Once the model is trained and the word embedding vectors were generated, a lower-dimensional representation can be obtained for each paper in the network by applying statistical operations over all the individual words extracted from the paper. Algorithm (3) shows an embedding-based algorithm for paper representation. The element-wise operator $\mathcal{T}(.)$ in Algorithm (3) can be obtained by applying different types of pooling techniques, such as maximum, minimum and average [14], to the words' embedding.

When studying larger text such as documents, there has been more active research on proposing novel ways for distributed representation. Beside the well-known Doc2Vec model, unsupervised document embedding has been also tackled in [74] using CNNs. The

Algorithm 3 Embedding-based Paper Representation

0: **Inputs:** Set of papers $\{p_1, \ldots, p_N\}$ and associated raw-text

0: **Parameters:** Size of dictionary (J), dimension of embedding (H) and training window-size (L)

0: **Outputs:** An $H \times 1$ vector representation of the paper

0: **Dictionary Generation:**

- **Form set of all the words from N papers as $\{w_1, \ldots, w_I\}$**
- **Apply word pre-processing steps (lemmatization, stemming, etc)**
- **Rank pre-processed words based on their frequency over the set of all the papers, choose top J words and form the dictionary $\{\bar{w}_1, \ldots, \bar{w}_J\}$**

0: **Training-data Generation**: Generate tuple of training instances by extracting L left and right surrounding words to each certain word w_m

0: **Model Training**: Train one of CBOW or SKG models using all the generated training tuples and obtain optimal weight matrix W^* and bias vector $\mathbf{b}^*$ of the hidden layer

0: **Embedding Generation**: For each word in the dictionary, calculate the embedding vector using the trained model

0: **Paper Vector Representation:**

for i in N **do**

Extract all distinct words, apply lemmatization and stemming, and form the set $\{\bar{w}_1, \ldots, \bar{w}_{N_i}\}$

Form the set of embedding vectors $\{\mathbf{e}_1, \ldots, \mathbf{e}_{N_i}\}$ by applying any of the distinct words to the trained model

Obtain the paper vector representation by applying the element-wise operator $\mathcal{T}(\mathbf{e}_1, \ldots, \mathbf{e}_{N_i})$ to the set of words' embedding

end for=0

proposed model in [74] extracts textual data and segments each text into two chunks of input/output. The network then aims to form an embedding for the input text and, then, predicts the output sequence using another stack of the network. It has been shown that the proposed technique outperforms many existing document representations such as Doc2Vec in tasks such as IMDB sentiment classification. In [75], the original Doc2Vec model has been improved to a new model called Doc2Vec through Corruption (Doc2VecC). The main idea in [75] is to regulate the training objective in Doc2Vec to favor informative or rare words and enforce common words to admit zero embedding. The new model has been tested against the original Doc2Vec and several other methods on some standard classification and sentiment analysis tasks. Other approaches in [76, 77, 78] have also proposed novel ANN-based structures to project document text into a lower dimensional vector.

2.1.2 Citation Based Representation

Unlike the content-based representation techniques that emphasize on the textual content of each document, the idea behind citation-based methods is to represent each document based on its relation with other documents in a citation graph.

A large citation graph can be usually represented as a sparse adjacency matrix where each entry denotes the existence or relation weight between each two nodes in the graph. In this case, each node can be represented as a large-dimensional vector of relation among other nodes. Given the curse of dimensionality, techniques such as matrix factorization [84] have been widely used in the literature to project the document vector representation to a lower-dimensional space. A complete survey of citation-based methods for document representation in a graph can be found in [85]. It has been shown in [85] that the matrix factorization can be applied to the adjacency matrix to obtain a lower-dimensional node representation using different techniques such as sparse SVD [86].

Collaborative filtering [87] has been also widely used in the literature to represent a document within a citation graph and, then, find similar documents for the purpose of recommendations [88, 89] and document clustering [90]. The application of deep learning in collaborative filtering has been also studied in [91, 92]. The main motivation for using deep learning is to tackle the commonly known cold-start problem that appears in common collaborative-filtering techniques due to the emergence of new documents that

have no citation information. In this case, an ANN architecture aims to admit a local node vector as the input and learns the latent representation through some existing training data such as user interests and reviews. In [92], it was shown how a generative ANN can be used in projecting each document to a lower-dimensional space with the model being trained on the user-data such as co-clicks and co-views. It has been shown that the given representation outperforms many local matrix-factorization based techniques in tasks such as movies recommendation.

While most literature has focused on building efficient models that produce representative vectors of documents using either text or citation-information, they do not still propose a systematic way to integrate the citation information into the process of generating text representation and, then, propose a fully hybrid technique. The next section reviews related work on using distributed learning for hybrid representation of the text.

2.2 Related Work

Distributed representation of documents using their relationship against other documents through a citation graph has been studied in the literature through adopting the seminal Word2Vec method [79]. In [79], a novel approach was proposed to construct document embedding in large citation networks. The Large-scale Information Network Embedding (LINE) technique proposed in [79] defines a first and second-order proximity between each pair of nodes as a probability measure. Then, an objective function is defined to maximize the similarity of adjacent nodes given the definition of the proximity. The optimizer then projects each node (document) to a lower-dimensional representation.

Online learning of social representations has been conducted in [80] using DeepWalk. The algorithm in [80] inherits the well-known skip-gram model proposed in [18, 81] for Word2Vec model where each node in the graph is represented by its relation to neighbor nodes. DeepWalk then proposes a novel Random Walk model to generate training tuple for the skip-gram model and, then, an ANN is trained using the generated tuple. The lower-dimensional representation of each document can be then generated by the output of the ANN. It has been also shown in [80] that the algorithm outperforms the majority of state-of-art node embedding techniques such as Spectral Clusterring [82] in many supervised tasks like multi-label classification and link prediction.

While the methodology in [80] produces neighbors of each vertex in a graph through a first-order random-walk model, the newly established Node2Vec [27] forms training tuple through a novel second-order random-walk model [83] . The main idea in [27] is to use a combination of Breadth and Depth-first search across the graph to sample neighbors of each node in the graph. A Skip-gram model is then used to project each node to its neighbors and form an embedding vector for any node in the graph. Experimental results in [27] show that the Node2Vec approach beats many other techniques such as DeepWalk in tasks like classification and link prediction. As the approach is forming training tuple as a mixture of both Depth and Breadth-first searches, it can be observed that Node2Vec is able to represent documents in a more informative way since a more diverse set of neighbors is included in the training regime.

While both DeepWalk and Node2Vec provide a systematic way to produce lower-dimensional document representations using the citation graph, they suffer from two main drawbacks. First, both techniques rely on static graphs with no changes in the number and structure of nodes. Once a new node is added to the graph, none of presented techniques are able to project the node to a lower-dimensional representation. In addition, both techniques are unable to handle the cold-start cases where most recently added documents have not yet received citations and their representation through the pure citation relationship is not accurate. This can be a critical issue in the modern scientific graphs where scientific manuscripts are often added to the graph and using only-citation information may not suffice.

The proposed approach in this thesis is closely related to both DeepWalk and Node2Vec through adopting a second-order RandomWalk model to generate training tuple across the graph. However, this thesis proposes a new version of the distributed learning on the graph called content-based Node2Vec where the algorithm aims to build a hybrid distributed learning of graphs by adapting the traditional Node2Vec approach to admit document textual information. Unlike the Node2Vec architecture, our proposed technique admits the textual information using bag of words model and, then, projects each document to a lower-dimensional representation using an ANN. In the training phase, the embedding vector is used to generate bag of words vectors of the neighbor documents that have been produced by a second-order RandomWalk. Compared to both the DeepWalk and Node2Vec techniques, the content-based Node2Vec can handle citation graphs with a varying number of nodes. Once a new node is added to the graph, the

ANN received the bag of words vector of the generated document and projects it to a lower-dimensional embedding vector. In addition, for documents with no citation in the graph, as our approach uses the textual content of each node, a node representation can be produced based on a pre-trained model over the existing vertices in the graph. In this case, the algorithm aims to map the document to an embedding space where its distance to the documents with the similar textual documents is minimized.

2.3 summary

In this chapter, A literature survey was conducted on techniques for document representation. we discussed both content and citation based methods, and explained local and distributed alternatives for any of the aforementioned taxonomies. This chapter also reviewed TF-IDF as a local content-based technique and, then, presented topic modeling and word-embedding as two popular distributed approaches. The next chapter will go through the content-based Node2Vec as the main contribution of this thesis.

Chapter 3

Methodology

Considering a document as the main source of knowledge and a citation network as a source that shows how documents interact or correlate with each other, the following definitions are assumed:

Definition 3.1. *Let v_i represent the i-th document in an assumed scientific network of N manuscripts. Each document can be then represented by a collection of words $\{w_1, \ldots, w_{K_i}\}$ with K_i being the number of unique words in the i-th document.*

Definition 3.2. *Define $\mathcal{C} = \{V, E\}$ as a Directed Acyclic Graph (DAG) that represents citations among all N documents in the network where $V = \{v_1, \ldots, v_N\}$ denotes the set of vertices (aka documents) and $E = \{E_{ij}\}, i, j \in \{1, \ldots, N\}$ corresponds to the set of edges with $E_{ij} = 1$ if the i-th document cites the j-th one.*

A citation network defined as above may be mathematically represented as a big sparse $N \times N$ matrix with edges being non-zero entries of the matrix. The sparse representation of the citation graph can be then used for various analytical tasks such as recommendation and eigenfactor calculation [37].

Definitions in (3.1) and (3.2) reflect important aspects of information that could be extracted from scientific articles. While definition (3.1) presents each document as a major source of knowledge, it does not still provide any information regarding citation relationships among documents. The citation graph in (3.2) models users' feedback through directed edges from a citing manuscript to a cited document. Although such a directed relationship cannot be grasped from the main content of the document, citation

graphs still suffer from the lack of considering documents' content information that can be much valuable for many tasks such as predicting the eigenfactor of a newly published paper in the scientific domain.

Definition 3.3. *For a certain i-th document in above-defined scientific network, let $\boldsymbol{e}_i$ denote a $M \times 1$ vector of float numbers that represents the document. Also, define $f : d \rightarrow \boldsymbol{e}_i$ as a function that maps the document to its representation.*

The following criteria are assumed for any mapping function f and resulting representation vector:

- *Lower dimensional representation*: for computational purposes, it is more efficient that f maps documents into lower dimensional vectors that require less disk or memory storage. In addition, lower dimensional representation accelerates similarity metric calculation that is common in many recommendation tasks.

- *One-to-one mapping*: it is desirable that no two documents are assigned to the same representation. This indicates that the mapping function f should satisfy one-to-one condition. The aforementioned condition also implies that each document in the network is mapped to a unique vector representation.

Definition 3.4. *Given any certain word w_i drawn out of the set of all the documents $\{d_n\}$, we define $q : w_i \rightarrow \bar{w}_j$ as an arbitrary operator that produces a new word $\bar{w}_j$ where the operator q can be any of common lexical functions such as lemmatization or stemming [12].*

Assuming I as the total number of distinct words in the set of documents,the role of the operator q is to aggregate words with the same stems or lemmas and produce a smaller set of J distinct words.

Definition 3.5. *Define the word dictionary $\mathcal{D}$ as the set of M distinct words chosen from the list of all J words in the set $\{\bar{w}_1, \ldots, \bar{w}_J\}$ where M is chosen based on some filtering considerations (such as removing less-frequent words) to keep the size of dictionary more compact.*

In the following, our novel content-based Node2vec technique for document representation is presented.

3.1 Hybrid Distributed Representation of documents in the Citation Network

While each document can be individually identified in the scientific world through its textual content, citation relations among different documents within a graph can also indicate the presence of a contextual relationship between documents, something that cannot be necessarily obtained from the text data. These sets of information can be then used along with the documents' raw-text to learn a more universal lower-dimensional representation of scientific articles.

The Node2Vec method was first proposed in [27] to extract distributed representation for vertices in a large relation graph. The main idea is inspired by the traditional Word2Vec approach where it is assumed that each certain word, as the main basis of the text, within the text is semantically correlated with it surrounding tokens. The Node2Vec then customizes this idea to the relation graphs where each node can be characterized by its surrounding nodes. The main two drawbacks in the proposed algorithm in [27] are:

- The proposed methodology suggests creating a dictionary of the nodes for the whole graph and, then, designs an Artificial Neural Network (ANN) to project the local node representation to an embedding vector. This approach cannot be then scaled to the newly added nodes to the graph though. For example, in a massive citation network, if new documents are published and added to the citation graph, the algorithm cannot be properly re-formed to generate embedding for the new article.

- The algorithm in [27] does not consider the content of each node in the graph in the process of generating lower-dimensional representation of the node. Therefore, for cold-start documents (newly published documents with no citation), the proposed algorithm cannot be used as the document has not even appeared in the citation graph. In this case, the procedure in [27] is unable to generate a vector representation for the new document.

In this section, an extension of the Node2Vec approach is proposed to combine both content and citation information of scientific articles to generate a more accurate representation of documents. Unlike the technique in [27], our method can handle both above mentioned drawbacks by considering the raw text of each document in the process

of embedding generation. In the following, different steps in creating an architecture for distributed document representation using both content and citation information are discussed.

3.1.1 Record Generation and Pre-processing

The citation network for a scientific network was defined as a DAG in the last section. Here, we ignore the direction in the graph and consider the original graph C as an undirected graph where each edge represents a bi-directional relationship between a certain pair of documents (d_i, d_j). The reasoning behind this act is that in this experiment, the main focus is on the existence of a citation link between each pair of arbitrary documents regardless of the fact that which one is the citing paper and which one is the citing paper. Also, note that as only most-recent documents cite older articles, and not the opposite-way, such a change from the directed to undirected graph can capture the relationships among articles more realistically and help us to have a more enrich data per each document. To project each node to a lower dimensional vector, we need to consider each node in the context of its relation to the neighbors in the graph. In this case, each node and its neighbors form set of training data for an ANN that aims to map the node to a lower-dimensional vector representation. Figure 3.1 shows an undirected graph with a node u and its neighbors being highlighted. The neighbors of the node u can be approached using one of the following search patterns:

1. **Breadth-first Search**: all the first-level neighbors of the target node are visited in a Breadth-first Search (BFS). Figure 3.1 shows all visited vertices by BFS by a red arrow.

2. **Depth-first Search**: in a Depth-first Search (DFS), all the vertices falling in an arbitrary path initiating from u as the root-node are visited. All yellow arrows in Figure 3.1 show the vertices of a DFS in the graph.

Given the topological structure and the search type in an arbitrary graph, each node may be surrounded by a different-size set of neighbors. This will lead to both inconsistent and computationally extensive training procedure. To handle this issue, a fixed-length random-walk path of length L initiated from a vertex u can be defined as a set of L nodes

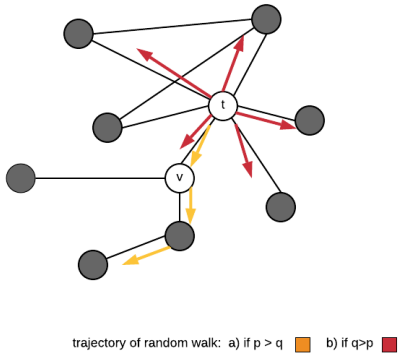


Figure 3.1: Different BFS and DFS search strategies in a graph. Red and yellow arrows show the strategies and their relationship with hyper-parameters p and q.

$\{c_1, c_2, \ldots, c_L\}$ whose transition probability can be given as follows [27]:

$$p\Big(c_i = x | c_{i-1} = v\Big) = \begin{cases} \frac{\pi_{vx}}{Z} & (x, v) \in E \\ 0 & \text{otherwize} \end{cases} \tag{3.1}$$

where E denotes the set of edges, π_{vx} is the un-normalized transition probability between pairs v and x, and Z is a normalization constant.

Definition 3.6. *For any triple (t, v, x) of vertices in graph C, define the following bias in the search path:*

$$b_{p,q}(t, x) = \begin{cases} \frac{1}{p} & \text{if } \mu(t, x) = 0 \\ 1 & \text{if } \mu(t, x) = 1 \\ \frac{1}{q} & \text{if } \mu(t, x) = 2 \end{cases} \tag{3.2}$$

where $\mu(t, x)$ denotes the shortest distance between vertices x and t and $\{p, q\}$ are search parameters that adjusts the trade-off between BFS and DFS search strategies.

The transition probability π_{vx} can be now calculated as follows:

$$\pi_{vx} = b_{pq}(t, x) w_{vx} \tag{3.3}$$

where w_{vx} denotes the static matrix weight that determines the linkage weight between any two arbitrary nodes in the network. The formulation in (3.2) allows the search guided

into both DFS and BFS paths. The transition probability in (3.3) can be interpreted as follows:

1. Current node is v and the previously visited node is known as t.

2. The search revisits t by a weight proportional to $\frac{1}{p}$. A high value of p increases exploration and avoids the search to revisit a previously sampled node. On the other hand, a low p encourages the search to stay around the generated nodes, which may suffer from lack of enough exploration.

3. The search visits a new node x that is 2 nodes away from t with a weight proportional to $\frac{1}{q}$. The parameter q can tune the trade-off between DFS and BFS. A large value of q ensures the algorithm rarely visits a deep node and, mostly, samples neighbor nodes (BFS). A small value of q pushes the search into farther nodes and explores deeper paths within the graph.

Algorithm 4 shows different stages in sampling L neighbor nodes for each certain document in the network.

3.1.2 ANN Architecture

Definition 3.7. *For each i-th document in the graph d_i, the m-th training record is defined as an $(L+1)$ tuple of documents shown as $r_m = (d_i, d_{n_1}, d_{n_2}, \ldots, d_{n_L})$ with L being the number of neighbor documents sampled (based on the algorithm in the previous section) and n_i denotes the index of the i-th neighbor in the graph.*

Definition 3.8. *The raw-feature vector for each i-th document within the citation graph is presented by a vector that shows the presence of each dictionary's word in the document. In this case, the i-th document is represented by a $M \times 1$ vector $\boldsymbol{I}^i$ whose m-th entry $\boldsymbol{I}^i_m$ is defined as follows:*

$$\boldsymbol{I}^i_m = \begin{cases} 1 & \textit{if } m\textit{-th word exists in the document} \\ 0 & \textit{otherwise} \end{cases} \tag{3.4}$$

The proposed structure in this section is based on the Skip-gram model shown in Figure 2.1 where each document is applied to an ANN and, then, aims to predict the

Algorithm 4 Neighbor Node Generation for Content-based Node2vec Model [27]

0: **Inputs:** Set of documents $\{d_1, \ldots, d_N\}$, undirected graph C constructed from the citation network

0: **Parameters:** Number of sampled nodes (L), return-parameter (p) and in-out parameter (q)

0: **Outputs:** A set of L neighbor documents associated with each node (document) in the citation graph

for i in N **do**

Let $c_0 = d_i$ and $l = 0$ represent the starting node and the node index in the neighbor set, respectively. Also, initialize $v = c_0$ and $t = NULL$

for l in L **do**

Find the set of neighbor nodes to v as $\mathbf{x} = \{x_1, \ldots, x_I\}$ with $(x_i, v) \in E$(set of edges)

For each node in $\mathbf{x}$, calculate the search bias value $b_{p,q}(.)$ using (3.2)

For each pair (x_i, v), calculate the transition probability $p(c_l = x_i | c_{l-1} = v)$ using (3.3)

Sample a node randomly from $\mathbf{x}$ based on the calculated transition probability and let $c_l = x_{i^*}$ with i^* being the index of the sampled node

Set $t = v$ and $v = c_l$

end for

Output $\{c_1, \ldots, c_L\}$ as the set of sampled neighbor nodes

end for=0

presence of words in the set of neighbor documents. Considering tuples of target documents and L neighbors generated by the procedure of Algorithm 4, the main components of the ANN model are as follows:

- Inputs: the input to the network is an $M \times 1$ vector that is created by eq. (3.4) for each target document.

- Projection layer: input $\mathbf{I}^i$ is projected to a lower-dimensional representation vector $\mathbf{z}^1$ through the following hidden-layer operation:

$$\mathbf{z}^1 = g\left(W^1\mathbf{I}^i + \mathbf{b}^1\right) \tag{3.5}$$

  where $\{W^1, \mathbf{b}^1\}$ are $N^1 \times D$ weight matrix and $N^1 \times 1$ bias vector, respectively, N^1 denotes the number of hidden variables, $g(.)$ is the sigmoidal function in the hidden layer, and $\mathbf{z}^1$ represents the embedding representation as the output of the hidden-layer.

- Output mapping layer: the embedding vector $\mathbf{z}^1$ is now mapped to an output probability vector that aims to predict the presence of words in the set of L neighbor documents. We define $\mathbf{y}^l_m$ as the probability of having the m-th word of the dictionary in the l-th neighbor document. The following soft-max function then produces the probability based on the output of the hidden-layer:

$$\mathbf{y}^l_m = \frac{\exp(W^2_{m:}\mathbf{z}^1 + \mathbf{b}^2_m)}{\sum_m \exp(W^2_{m:}\mathbf{z}^1 + \mathbf{b}^2_m)} \tag{3.6}$$

  with $\{W^2, \mathbf{b}^2\}$ being weight matrix and bias vector of the output layer, respectively, and $W^2_{m:}$ denoting the m-th row of $M \times N^1$ matrix W^2. Note that the form in (3.6) is similarly written for any of the neighbor nodes. The vector $\mathbf{y}^l$ then is formed with each entry corresponding to the probability of having each word of the dictionary in the l-th neighbor document.

3.1.3 Training and Application

To train the ANN discussed in the last section, we collect a set of training records $\{r_1, \ldots, r_T\}$ where each t-th record r_t consist of a target document and L neighbor

document.

Definition 3.9. *For each t-th training record r_t, the following cost function evaluates the probability of generating the words in L neighbor document using the ANN model:*

$$\mathcal{C}^t(\Theta) = \sum_{L} \sum_{m} \boldsymbol{I}^l_m \log \Big(p(\boldsymbol{y}^l_m | \boldsymbol{I}_{target}; \Theta) \Big) \tag{3.7}$$

where $\boldsymbol{I}_{target}$ denotes the input representation of the target document (eq. (3.4)), and $\boldsymbol{I}^l_m$ is a binary value corresponding to the existence of the m-th word in the l-th document. In addition, $p(\boldsymbol{y}^l_m | \boldsymbol{I}^l; \Theta)$ denotes a probability that measures the existence of the m-th word of the dictionary in the l-th neighbor document, and is calculated by (3.6). In 3.7, Θ represents the set of network parameters (weight matrices and bias vectors). The cost function for the whole ANN structure can be defined as follows:

$$\mathcal{C}(\Theta) = -\frac{1}{T} \sum_{t} C^t(\Theta) \tag{3.8}$$

The training part aims to minimize the cost function in (3.9) using all the T generated training records. The training can be then done by applying common gradient techniques such as Stochastic Gradient Descent (SGD) to the cost function in (3.6). For computational considerations, the training set can be also divided into L disjoint batches of records as $\{b_1, b_2, \ldots, b_L\}$ where all T training records are equally distributed among L mini-batches, and the SGD update can be done on each batch [14]. After training the network, each document in the citation graph can be now mapped to an embedding using the hidden-layer projection in (3.5). Note that the above representation remedies issues of algorithm in ([27]) by:

- Providing an architecture that takes document's content as the input and projects the local word-count representation to embedding.
- Making the structure responsive to newly added documents to the network as the proposed structure accepts raw-text instead of document-id.
- Extending the algorithm to handle cold-start documents through finding and embedding that could be also used to find the most similar existing documents in the

citation network.

A summary of stages for implementing the content-based node2vec algorithm on the collection of documents can be found in Algorithm 5.

Algorithm 5 Content-based Node2vec for Document Representation

0: **Inputs:** each of documents in the set $\{d_1, \ldots, d_N\}$ and associated text, and undirected graph G constructed from the citation network

0: **Parameters:** size of documents (N), size of dictionary (M), size of sampled nodes (L) and dimension of embedding (H)

0: **Outputs:** An $H \times 1$ vector representation of the document

0: **Dictionary Generation: build dictionary D of M words following the procedure in Algorithm 1**

0: **Training-data Generation**: Generate tuple of training instances by running Algorithm 4

0: **Model Training**: Apply a gradient rule over the objective function (3.9) and obtain optimal weight matrices $\{W^1, W^2\}$ and bias vectors $\{\mathbf{b}^1, \mathbf{b}^2\}$

0: **Embedding Generation**: For each document p_i:

- Create the input document vector I^i using the associated raw-text and by the procedure given by (3.4)
- Calculate the embedding vector $\mathbf{z}^{1,i}$ using (3.5)

0: **document Vector Representation:** present $\mathbf{z}^{1,i}, i = 1, \ldots, N$ as the set of lower-dimensional vector representation of documents =0

3.2 Computational Complexity

In this section, an overview on the computational complexity of each algorithm is presented. For each algorithm, the computational time is calculated considering both the pre-processing cost and the training time, if the algorithm requires parameter tuning. The complexity of each algorithm is analyzed as follows:

- **TF-IDF**: based on the procedure given in Algorithm 1, since the TF-IDF does not require any training phase, the order of complexity is obtained by calculating the time of building the representation vector for each input document. This equals to the time to build the TF-IDF vector for all the documents and can be found

by O(TF-IDF)= $O(NM)$ with N and M being the number of documents and the dimension of the dictionary, respectively.

- **Embedding**: The order of complexity can be divided to the complexity for training algorithm in the embedding model and the time required for preparing one-hot word vectors. The overall complexity is written O(embedding)= O_{EMB}(training)+ $O(NM)$.

- **LDA**: the LDA algorithm also requires time for both training and data-preparation. The orger can be written as O(LDA) = O_{LDA}(training)+ $O(NM)$.

- **Node2Vec**: order of complexity here consists of two components. First, complexity of the algorithm for generating neighbors according to Algorithm 4 if found. Then, the final complexity is found by adding the training-time to the above-mentioned time. Finally, the time the algorithm needs for each output vector is added to the previous components and provides the overall complexity as O(Node2Vec)= $O(a^2N)$+ O_{N2V}(training)+ $O(N)$ where a denotes the average degree of the citation graph.

- **Content-based Node2Vec**: This technique is associated with an extra step compared to Node2Vec technique since this requires to build an initial word count vector. Therefore, the order of complexity is found by O(Content-based Node2Vec) = $O(NM)$+ O_{N2V}(training) + $O(a^2N)$.

Table 3.1 summarizes each algorithm along side with their calculated order of complexity. Note that N and M are representing, respectively, the number of all documents in the graph and the size of dictionary, similar to the previous sections. Also, a is a score declaring the average degree of the graph which means how many neighbors are averagely linked to each single node in the citation graph.

3.3 Summary

Content-based Node2Vec approach was discussed in this chapter as a hybrid technique for the vector representation of documents using both the content of documents and citation graph. An MLP neural network was proposed to produce the vector representation

Algorithm	Complexity
TF-IDF	$O(NM)$
Embedding	$O(NM) + O_{EMB}$(training)
LDA	$O(NM) + O_{LDA}$(training)
Node2vec	$O(a^2N)+ O_{N2V}$(training)$+ O(N)$
Content-based Node2vec	$O(a^2N)+ O(NM)+ O_{N2V}$(training)

Table 3.1: Order of complexity of all the algorithms discussed for document representation.

of each document using its textual data. To train the neural network, the neighbor documents to each document in the citation graph were generated using a second-order random walk model. Then, an objective function was defined to measure the capability of the neural network in predicting the content of neighbors for each input document. The training algorithm for the proposed structure was derived and the computational time of the content-based Node2Vec was compared to other techniques discussed in this thesis for document representation. Next chapter will discuss the application of the proposed method in the link-prediction problem along with a comparison to other state-of-art document representation approaches.

Chapter 4

Experimental Results

The performance of all the techniques discussed in this thesis for lower-dimensional representation is evaluated by considering the link-prediction problem on a certain experimental data-set. The data-set is prepared by following these steps:

- **Data type and pool-size**: the textual data is extracted from the data-base of scientific articles in the field of Computer Science. For this experiment, a subset of $109,000$ papers is sampled from the pool of all computer-science related papers available in Microsoft Academy (MSA) database. To sample the papers, a list of top-50 conferences and journals in the computer science is created and all the papers published in the list of above conferences and journals over the last 10 years are taken into consideration.

- **Raw-data generation**: the raw-data in this experiment is generated from two different sources. The abstract for all the papers in the pool is extracted from the MSA database [1]. It is also worth to mention that due to the restricted access to the full text of the papers we ended up using the abstract of the papers. Beside the abstract data, the MSA network also provides the citation network for all the papers in the database. For the purpose of this experiment, only those citation records where both citing and cited articles exist in the paper-pool are considered. For the set of $109,796$ papers, the final citation network results in $146,819$ records.

[1] https://labs.cognitive.microsoft.com/en-us/project-academic-knowledge.

Table 4.1 shows the general specifications of the data-source considered for the experimental results in this paper.

Number of Abstracts	$109,796$
Number of Citations	$146,819$
Average Size of Abstracts	$147.13(words)$
Average Number of Citations	2.8

Table 4.1: General specifications of the raw-data created for experimental results.

4.1 Vector Representation

The content of each paper is used as the input to the ANN structure to produce the vector representation. The ANN training is done using the citations among documents where the content of each document is embedded based on its relationship with other documents across the citation graph. The data-preparation step is conducted as follows:

- **Tokanization**: a standard white-space tokanizer is applied to all the abstracts, each token is lower-cased, and all the digits are masked from the tokens.

- **Data enrichment**: to enrich the data-set and aggregate the words with same stems and lexical meanings, each token is both stemmed and lemmatized.

- **Dictionary Generation**: the tokens are ranked based on the term-frequency within all the abstracts. Due to computational complexity, the aim is always to while preserving the most frequent words(most important words), keep the size of the dictionary as compact as possible. For this reason, we selected the value of threshold such that all the tokens that construct the majority of the total mentions (in the of text of all documents) are preserved and added to the dictionary list. This can be done by calculating the histogram of the tokens' frequencies and, then, choose the threshold such that 95% of total frequencies covered by all the tokens that pass the threshold. This will result in a dictionary of size M. For this experiment, $\tau = 8$ is chosen and a dictionary of $M = 9078$ tokens is created.

Given the created dictionary, any of the following techniques is applied to generate a lower-dimensional vector representation:

- **TF-IDF**: both TF and IDF values are calculated for the dictionary tokens. Each paper in the pool is then represented by a $M \times 1$ TD-IDF vector.

- **Topic-model**: the LDA algorithm is now applied to the abstract data to find the vector representation of each paper. The number of topics is also chosen to be $K = 50$ as mentioned in [15] as a reasonable number of topics for general document representation problems. Note that other more advanced techniques can be also used to find the optimal number of topics whose details can be found in [93, 94]. As the set of textual data used in this thesis has similar lexical content to what being experimented in [15], a choice of $K = 50$ was made for the experiments. Both prior hyper-parameters α and β are to be uniformly initialized across number of topics and size of the dictionary. The LDA model is trained on all the abstracts using Collapsed Gibbs sampler [36]. Using the trained model, each paper is projected to a $K \times 1$ of topical weights.

- **Word-embedding**: the Cbag of words model is trained on the abstracts using the steps given in Algorithm 3 where $L = 2$ (size of surrounding words) and $H = 100$ (dimension of embedding) are considered as values for the hyper-parameters. Once the embedding vector is found for all the tokens in the dictionary, the embedding vector for each paper is found by calculating an average of all individual tokens' embeddings.

- **Hybrid Approach**: training records are generated based on the surrounding papers of each certain manuscript in the citation graph. The hyper-parameters p and q in Algorithm 4 are tuned based on the supervised task where paper embedding is used as the feature vector. The other parameter L (size of neighbor-set) is assumed to equal 10. Given each node in the graph, a set of 10 neighbor nodes is chosen based on the algorithm in Algorithm 4. The size of hidden-layer H is also decided to be 100. Note that the choice of L or H can be itself a hyper-parameter selection problem. However, it has been shown in [27] that a value of $L = 10$ is a proper selection for the size of neighbours set and also $H = 100$ has produced the best word vector representations in [18]. Since the dictionary of words created for the

set of papers is a subset of the actual words being considered in [18], the same value of $H = 100$ was also chosen for the experiments in this section. The Skip-gram model is now trained using all generated records and once the model is trained, the embedding vector representation is generated for all the papers in the network.

4.2 Link-prediction Problem

The procedure discussed in the last section is used to extract feature vectors for each paper in the manuscript. To evaluate the efficacy of the generated vectors, each method is tested against predicting the existence of a link (edge) between any two nodes (papers) in the citation graph. For this prediction problem, the direction of the edge is ignored and the existence of a citation is considered as a relation between two certain papers.

The link-prediction problem can be now formulated as a binary classification scenario as follows:

- **Feature-space**: the set of links (edges) form the space of features in the link-prediction problem. Having the vector representation of each node as a feature-vector, the feature for each edge is created by considering both the **average** and **Hadamard** operations (element-wise vector multiplication)over the embedding vector of any of the two nodes forming the edge.

- **Negative instances**: negative instances (no-link between two nodes) are generated by sampling disconnected nodes in the citation graph. To do this, for each given positive label and associated vertices $\{n_1, n_2\}$ with an existing edge, 10 negative nodes are randomly sampled from the network so that for each newly sampled n'_j, there is no direct link to either n_1 or n_2. Note that this ratio has been selected experimentally and after testing a few ratio, we realized any value above 10 does not make a considerable difference in the results. Considering the higher computational complexity associated with higher value, the value of 10 seemed to be reasonable. Given the size of the graph and number of citations, the above procedure generates $146,306$ positive instances and $1,463,023$ negative labels.

- **Binary classifier and performance metrics**: the link-prediction problem can be now modeled as a binary classification scenario. A logistic regression [35] classifier

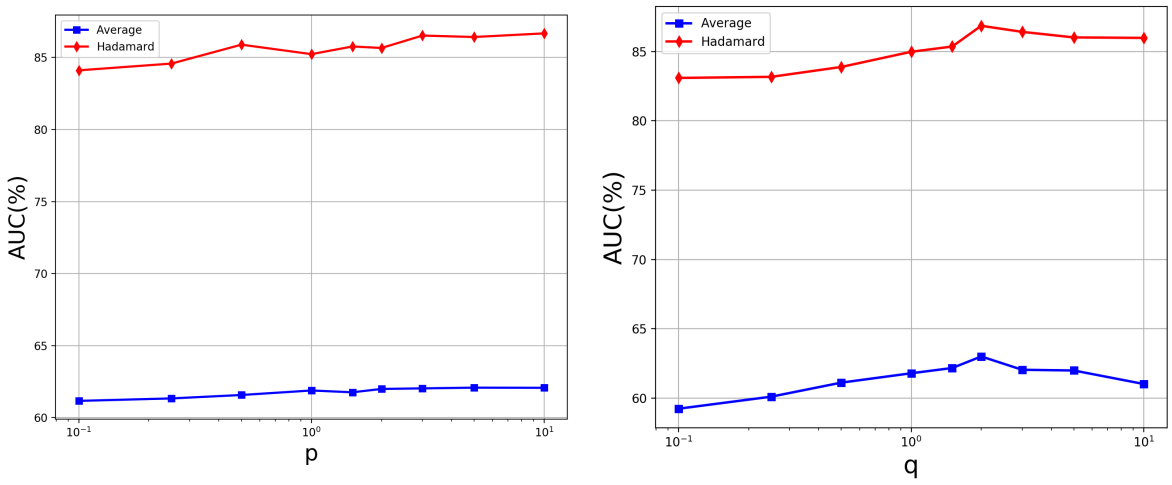
is used to undertake the task of binary classification. For performance evaluation, a 10-fold validation approach is used where 90% of data is used for training and the last fold is reserved for metric calculation. Given all the scores in the test-set, the Area Under Curve (AUC) is calculated by, first, depicting the number of true versus false positives and, then, calculating the area for the given curve.

The logistic regression classifier is first applied using the feature vectors obtained by the Node2vec model. To calculate hyper-parameters p and q, the Skip-gram model is trained using different values of p and q and the classification performance is computed for each certain case. Figure 4.1 shows the AUC metric computed for any values of hyper-parameters and, also, both choices of feature-vector generation for the edges. The results in Figure 4.1 indicate that:

- The scenario with Hadammard-generated edge-features produce around 20% higher metric in terms of AUC. Note that while other classification metrics such as accuracy could be used here, the AUC is preferred given the imbalanced nature of our dataset. Given the imbalanced ratio of 10, AUC can evaluate the performance of each technique more properly than other metrics such as accuracy. This performance gap also seems uniform over all values of hyper-parameters.

- The results in Figure 4.1 show negligible performance change when the value of p varies. Diving deeper into the results, the maximum difference in the metrics with a change in p is around 1%. Note that varying values of p urges the walk to move from local nodes around the starting point to farther paths. As the current citation network does not show very deep paths, such a conclusion was also expected.

- Unlike p values, the performance shows higher sensitivity to the other hyper-parameter q. When the value of q starts to increase, the performance metric also shows an increase pattern reaching to a 4% jump at $q = 3$. The metric then slightly decreases when the value of q starts to increase again. As low and high value-regions of q represent DFS and BFS patterns, respectively, this experiment shows that a moderate value of q that combines both BFS and DFS strategies provides the best performance metric. Given the insensitivity of the metric values to p and the shallow topology of the network, the above conclusion also looks intuitive.



(a) a

(b) b

Figure 4.1: Impact of hyper-parameters on the performance of node2vec features for link-prediction, (a) impact of return-parameter p (for $q = 1$) and (b) impact of in-out parameter q (for $p = 3$).

Given the best set of hyper-parameters p and q, the performance metrics for the node2vec method are now compared to other feature-extraction methods discussed in this manuscript. Table 4.2 shows the AUC metrics using all the feature-extraction techniques. Also, Figure 4.2 show the empirical Receiver Operating Curve (ROC) for any of the algorithms. To produce features for each edge, both Hadamard and average operators are used and the performance of the model under any of above techniques is evaluated. Note that Hadamard product can implicitly model other types of statistics such as min and max and, therefore, is used in many other link-prediction problems [27]. It is evident from the results that the Hadamard operator gives the best performance in all cases with an average of 17% higher AUC compared to the average operator. Comparing all the individual representations, the following observations are made:

- The TF-IDF shows the worst performance in terms of the AUC. The results for TF-IDF shows around 15% lower performance when compared to Hybrid Node2Vec.

- The performance gap between LDA and TF-IDF seems negligible. For Hadamard operation, LDA shows around 4% higher AUC than TF-IDF, but still 10% lower

than Hybrid Node2vec.

- Embedding method outperforms both LDA and TF-IDF with 8% higher AUC in both Hadamard and average operations. Indeed, representing each paper through its individual words' embedding vectors has brought a significant value to the final performance of the classifier.

- In both Hadamard and average cases, the hybrid Node2vec shows a significant performance improvement over both TF-IDF and LDA methods. Compared to the embedding base-line, the approach proposed in this paper shows $3-6\%$ AUC improvement depending on the type of edge-feature operation.

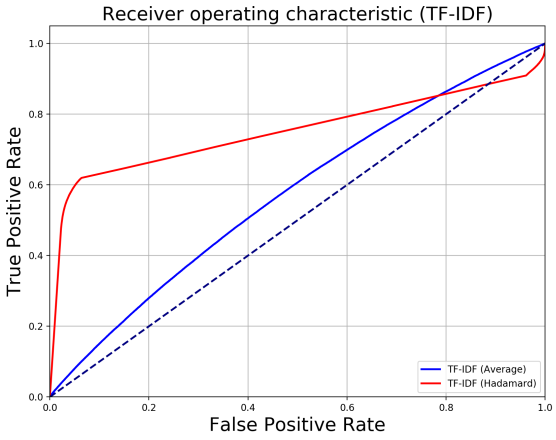
Metric	ALG		ALG		ALG		ALG	
	TF-IDF		LDA		Embedding		Hybrid Node2Vec	
	Avg	Had	Avg	Had	Avg	Had	Avg	Had
AUC(%)	55.48	71.92	56.6	75.15	59.082	80.86	62.019	86.52

Table 4.2: Area Under Curve (AUC) metric for link-prediction using any of four techniques presented in this paper. All the numbers are computed as the average over 10 rounds of experiment

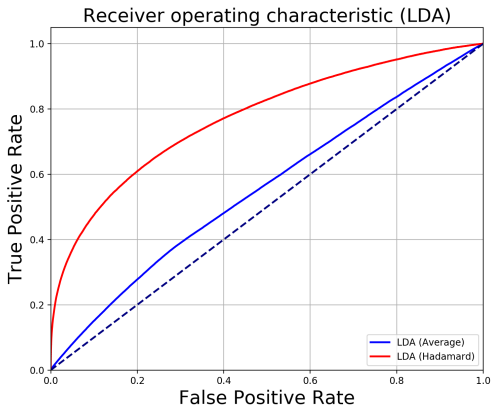
4.3 Distribution of Prediction Scores

Table 4.2 presented the results for the edge-prediction problem in terms of AUC. It is also informative to study the distribution of generated scores by the logistic regression classifier as it will show the effectiveness of each technique in identifying the presence of an edge in the graph. We extract the scores for both the negative and positive instances and all the algorithms discussed in this section. The empirical distribution of scores is now depicted in Figure 4.3 where results are shown for negative and positive instances separately.

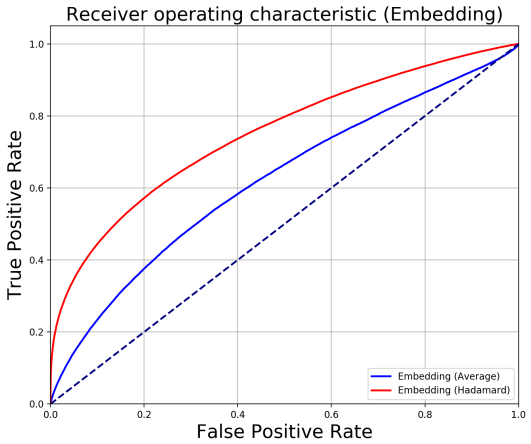
The distribution of positive instances shows two spikes at low and high scores when TF-IDF outputs are considered. Note that due to the higher dimensionality of the TF-IDF representation, the logistic regression classifier projects all instances to either low or high score regions. Unlike TF-IDF outputs, all three other approaches show a



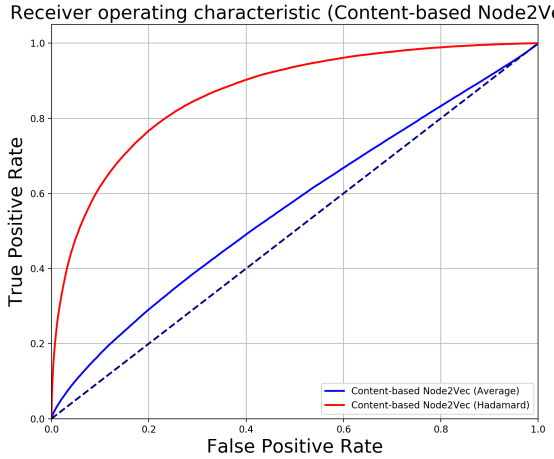
(a)



(b)

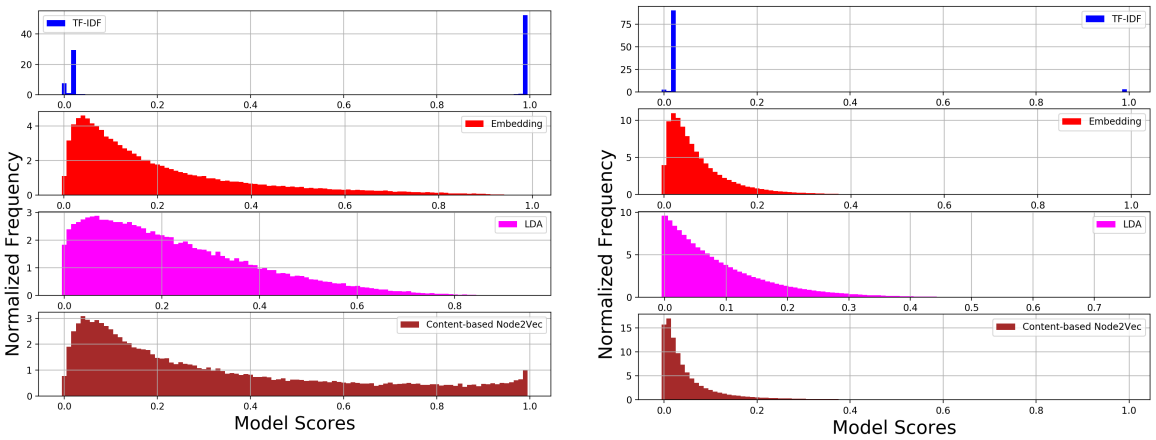


(c)



(d)

Figure 4.2: Receiver operating curve for (a) TF-IDF, (b) LDA, (c) Embedding, and (d) Content-based Node2Vec algorithm. The graph have been depicted based on the scores generated by single round of experiment.



(a) Positive Labels

(b) Negative Labels

Figure 4.3: Empirical distribution of generated scores by the model and for any of techniques used in this chapter for edge-prediction, (a) Distribution of scores for the positive labels, (b) Distribution of scores for the negative labels.

continuous distribution with a strong mode in lower score regions, which is expected due to the imbalanced nature of dataset and the model's bias in projecting inputs to negative labels. From Figure 4.3, it can be observed that the Hybrid (Content-based) Node2Vec approach projects a higher portion of instances to the higher-score regions, especially, when compared to the LDA method with the majority of scores accumulated around the mode. Calculating the mean and variance of distributions, it can be also observed that the Hybrid Node2Vec gives the highest mean (.33) while LDA stands after Embedding with (.23) as the mean of scores.

The graph in the right-hand side of Figure 4.3 shows the distribution of scores generated by the model and for the negative instances. Ideally, the model is expected to project majority of negative instances to lower-score regions. It is observed that the Hybrid Node2Vec method outperforms other techniques by producing an empirical distribution of scores (see Figure 4.3) that is heavily skewed to near-zero values. Compared to other techniques, the results by Hybrid Node2Vec show both lower mean and variance that implicitly indicates the capability of the technique in separating negative and positive instances.

Note that the results in Figure 4.3 can be also related to the AUC metrics presented

in Table 4.2. The raw scores generated by each model can be mapped to predicted labels by applying a threshold to each given score. For example, setting a score threshold for the scores in Figure 4.3, all the instances with a generated score above the threshold are considered as positive. Therefore, both the true and false positive rates can be immediately calculated. As the AUC is found by calculating the area under the curve of true vs. false positives, a higher AUC usually means a higher/lower true/false positives and can be, then, related to the distribution of raw-scores. As the raw scores' distribution for Hybrid Node2Vec in Figure 4.3 show higher mean for positive labels and lower mean and variance for negative instances, it is also expected that the scores can result in higher/lower rates of true/false positives when the threshold sweeps the interval $[0, 1]$.

4.4 Notes on Space and Time Computations

Beside the performance evaluation through accuracy calculations, it is essential to analyze the efficiency of any of the presented techniques in this thesis in terms of both time and required storage. The analysis here is done based on the time and space required for generating paper embedding vectors and, therefore, the classification time is not taken into consideration.

- **Storage**: the storage is determined by the space requires for storing all the papers' representation. We assume a data-base with N papers where each paper is represented by a L_A-length vector of 8-byte float numbers where A determines the algorithm name. The total space required for data-storage is then calculated as follows:

$$S_A = N \times L_A \times 8 \tag{4.1}$$

  It is then observed that the space requirement varies linearly with respect to the size of the embedding. While all embedding, Hybrid Node2vec and LDA approaches provide almost equal representation dimensionalities, TF-IDF approach gives a much larger vector calculations. For the simulation results generated in this paper, TF-IDF has almost 100 times larger vectors, which results in the same higher storage requirements compared to other techniques.

- **Computational Time**: all the results in this thesis are generated on an Amazon

m4-xlarge EC2 instance with 4 CPU cores and 16GB of RAM. All embedding generation and ANN training stages are also done in TensorFlow. Table 4.3 shows the calculated timing performance in seconds after averaging over 5 different runs where the time taking for word pre-processing (lemmatization and stemming) has been ignored. From the results, it can be observed that TF-IDF method requires much less computational time compared to other approaches with almost 100 times faster operation to the Hybrid Node2vec method proposed in this thesis. LDA shows 10 times slower than TF-IDF but has 10 times faster computational time in comparison to the Hybrid Node2vec method.

Method	**Representation Generation Time**
TF-IDF	$223.67(s)$
LDA	$2,064.0(s)$
Embedding	$18,723.0(s)$
Hybrid Node2Vec	$26,583.4(s)$

Table 4.3: Computational time performance of all the techniques used for extracting lower-dimensional representation vectors.

4.5 Summary

This chapter presented link-prediction problem as a benchmark to compare the performance of different techniques for document representation. We used the content and citation information of a selected subset of papers from top journals and conferences in the field of computer science. The content-based Node2Vec approach and other discussed methods were applied to the above-mentioned dataset to produce document vector representation. A logistic regression classifier has been also used to deal with the link-prediction problem using document vectors as the input features. Using the area under curve, the performance of all the algorithms was compared and the computational complexity of each algorithm was also discussed.

Chapter 5

Conclusions and Future Works

Representation of textual documents has been an active line of research in the NLP community due to its importance and application in many fields such as classification and clustering, recommendation, question-answering and web-search. The research area has been under development for several years from people have been using traditional Bag of Words (bag of words) and TF-IDF representations to deal with the large collection of textual data. With emergence of ANNs and deep learning, the above area of research has made a significant transition to the area of distributed representation learning from the available data sources.

This thesis made a comprehensive overview on the existing techniques for the document representation in the NLP domain. The thesis mainly made its analysis and review based on the recent categorization of research in this area into local and distributed representations of data that can be words or documents. Different families of local methods such as TF-IDF and bag of words models were discussed. In addition, the thesis reviewed state-of-art in document representation in the presence of a relational citation graph and discussed distributed techniques in the area.

The distributed representation has been the main focus of this thesis. The most recently developed content-based distributed representation techniques such as topic-modeling and word2vec-based methods were first explained. The application of ANNs in using the citation graph as the main data-source and, then, producing lower-dimensional document representation was presented by discussing techniques such as Node2Vec. The thesis also explained the drawbacks and possible contributions that could be added to

the recently developed approaches.

As a main contribution, this thesis proposed content-based Node2Vec as a hybrid distributed technique for document representation in the large corpus of textual data. The original Node2Vec model was modified to admit the documents' text as the main input and a Skip-Gram model was then employed to generate document embedding using an ANN as the basic parametric model. It was also shown that the new technique can handle many common drawbacks in the non-hybrid methods such as cold-start problem through building a system that can produce the vector representation using the document's text, which could admit the newly published document's text as the input and produce the representation.

The novel algorithm in this thesis was also applied to the link-prediction problem in the citation graph of scientific literature. A large collection of manuscripts in the field of computer science were used as the experimental data where both the textual content and the citation graph were used as the knowledge sources. Results in this thesis verified that the proposed technique in this thesis outperformed both the local and traditional distributed representation techniques such as word embedding and topic-modeling based techniques. In addition, the time-efficiency of the algorithm was compared against the traditional techniques and its superiority was verified through empirical results.

5.1 Future Works

The content-based Node2Vec approach proposed in this thesis can be extended and analyzed in three different directions. First, the technique can be improved by using more sophisticated ANN structures such as deeper architectures that could potentially model more complicated semantic content of the data. While the thesis proposed using a simple MLP network, other architectures such as CNN or recurrent ANNs can be also used to project the document to a lower-dimensional representation vector. In addition, beside the raw textual content of each document, other sources of information such as the full-text of the document and citation context may be used in the training phase and as the input to the network.

As another extension, more rigorous analysis and application of the proposed technique can be considered in different domains such as recommendation and question-

answering. As the content-based method can admit the documents' raw-text as its input, its superiority in dealing with cold-start problem can be discussed and verified by applying the technique to a standard recommendation problem. In addition, the approach can be used in many domain-specific web-search and question-answering where the document representation can be used as the base document-embedding being used in a supervised/unsupervised task.

Finally, while the algorithm in this thesis was mainly applied to the text domain, an extension to other sources of data such as images is desirable. In this case, each node within a citation graph may be represented by both the text and images and using an images integrated with the textual data as the input to the network is one possible extension. Also, the algorithm can be adopted to admit citations between different types of vertices such as text and images. In this case, the Node2Vec algorithm can produce universal document representations that embed both different modalities of the data and have learned different kinds of citation relationships.

Bibliography

[1] N. Shah and S. Mahajan, "Document clustering: a detailed review", International Journal of Applied Information Systems, Vol. 4, No. 5, October 2012.

[2] G. Manimekalai, K. Sathiyakumari, and V. Preamsudha, "A survey on various approaches in document clustering", International Journal of Computer Technology Applications, Vol. 2, No. 5, pp. 1534-1539, October 2011.

[3] C.C. Aggarwal, and C. Zhai, "A Survey of Text Classification Algorithms". In: Aggarwal C., Zhai C. (eds) Mining Text Data. Springer, Boston, MA, 2012.

[4] T.N. Rubin, A. Chambers, and P. Smyth, "Statistical topic models for multi-label document classification", Machine Learning, Vol. 88, pp. 157-208, 2012.

[5] S. Tong and D. Koller, "Support vector machine active learning with applications to text classification", Journal of Machine Learning Research, Vol. 2, pp. 45-66, 2001.

[6] Q. Shen, T. Wu, H. Yang, Y. Wu, H. Qu, and W. Cui, "NameClarifier: a visual analytics system for author name disambiguation", IEEE Transactions on Visualization and Computer Graphics, Vol. 23, No. 1, 2017.

[7] S. Cucerzan, "Large-Scale named entity disambiguation based on wikipedia data", Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 708716, Prague, 2007.

[8] T. Atapattu, K. Falkner, and N. Falkner, "A comprehensive text analysis of lecture slides to generate concept maps", Computers and Education, Vol. 115, pp. 96-113, December 2017.

[9] N. Kunaver and T. Pozrl, "Diversity in recommender systems a survey", Knowledge Based Systems, Vol. 123, pp. 154-162, 2017.

[10] Y. Lee, J. Leom, K. Song, J. Ha, and S. Kim, "Recommendation of research papers in DBpia: a hybrid approach exploiting content and collaborative data", IEEE Conference on Systems, Man and Cybernetics, Budapest, Hungary, 2016.

[11] J. Bobadialla, F. Ortega, A. Hernandez, and A. Gutierres, "Recommender systems survay", Knowledge-based systems, Vol. 46, pp. 109-132, 2013.

[12] C. Manning and H. Schtze, "Foundations of statistical natural language processing", MIT Press, Cambridge, MA, May 1999.

[13] S. Bttcher, C. L. A. Clarke, and G. V. Cormack, "Information retreival", MIT Press, Cambridge, MA, 2010.

[14] I. Goodfellow, Y. Bengio, and A. Courville, "Deep learning", MIT Press, Cambridge, MA, 2016.

[15] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent drichtlet allocation"", Journal of Machine Learning Research, Vol. 3, pp. 993-1022, 2003.

[16] M. Amami, G. Pasi, F. Stella, and R. Faiz, "An LDA-based approach to scientific paper recommendation", Natural Language Processing and Information Systems. NLDB 2016. Lecture Notes in Computer Science, Vol. 9612, 2016.

[17] C. Pan and W. Li, "Research paper recommendation with topic analysis", International Conference On Computer Design And Applications, China, June 2010.

[18] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of Word Representations in Vector Space", International Conference on Learning Representations, Arizona, USA, 2013.

[19] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality", Advances in Neural Information Processing Systems, pp. 3111-3119, Lake Tahoe, USA, 2013.

[20] M. Chen, "Efficient vector representation for documents through corruption", International Conference on Learning and Representation, Toulon, France, 2017.

[21] J. Kiros, Y. Zhu, R. Salakhutdinov, R. S. Zemel, A. Torralba, R. Urtasun, and S. Fideep learninger, "Skip-thought Vectors", Neural Information Processing Systems, Montral, Canda, December 2015.

[22] Q. Le and T. Mikolov, "Distributed representations of sentences and documents", Proceedings of 31st International conference on Machine Learning, Beijing, China, 2014.

[23] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: item-to-item collaborative filtering", IEEE Internet Computing, Vol. 7, No. 1, February 2003.

[24] J. Beel, B. Gipp, S. Langer, and C. Breitinger, "Research-paper recommender systems: a literature survey", International Journal on Digital Libraries, Vol. 17, No. 4, pp. 305-338, November 2016.

[25] K. Hashimoto, G. Kontonatsios, M. Miwa, and S. Ananiadou, "Topic detection using paragraph vectors to support active learning in systematic reviews", Journal of Biomedical Informatics, Vol. 62, pp. 59-65, 2016.

[26] C. Wang and D. Blei, "Collaborative topic modeling for recommending scientific articles", Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining, San Francisco, USA, 2011.

[27] A. Grover and J. Leskovec, "node2vec: scalable feature learning for networks", Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, USA, 2016.

[28] L. M. de Campos, J. M. Fernndez-Luna, J. F. Huete, and M. A. Rueda-Morales, "Combining content-based and collaborative recommendations: a hybrid approach based on Bayesian networks", International Journal of Approximate Reasoning, Vol. 51, pp. 785799, 2010.

[29] Y. Ding, G. Zhang, T. Chambers, M. Song, X. Wang, and C. Zhai, "Content-based citation analysis: the next generation of citation analysis", Journal of the Association for Information Science and Technology, Vol. 65, No. 9, pp. 1820-1833, June 2014.

[30] T. Achakulvisut, D. E. Acuna, T. Ruangrong, and K. Kording, "Science concierge: a fast content-based recommendation system for scientific publications", PLOS ONE, Vol. 11, No. 7, June 2016.

[31] H. Tian and H. H. Zhuo, "Paper2vec: citation-context based document distributed representation for scholar recommendation", https://arxiv.org/abs/1703.06587v1.

[32] S. Ganguly and V. Pudi, "Paper2vec: combining graph and text information for scientific paper representation", J.M. Jose et al. (Eds.): ECIR 2017, LNCS 10193, pp. 383395, 2017.

[33] M. Hoffman, D. Blei, and F. Bach, "Online learning for latent Dirichlet allocation", Proceedings of Advances in Neural Information Processing Systems, Vancouver, December 2010.

[34] T. Griffiths and M. Steyvers, "Finding scientific topics", Proceedings of the National academy of Sciences of the United States of America, Vol 101, pp. 52285235, 2004.

[35] K. Murphy, "Machine learning: a probabilistic perspective", MIT Press, 2012.

[36] G. Casella and E. I. George, "Explaining the Gibbs Sampler", The American Statistician, Vol. 46, No. 3, 1992.

[37] C. T. Bergstrom, "Eigenfactor: Measuring the value and prestige of scholarly journals", RL News, Vol. 86, No. 5, 2007.

[38] "PubMed: MEdeep learningINE Retrieval on the World Wide Web". Fact Sheet. United States National Library of Medicine, 2002.

[39] A. Sinha, Z. Shen, S. Yang, H. Ma, D. Eide and K. Wang, "An overview of Microsoft Academic Service (MAS) and applications", Proceedings of the 24th International Conference on World Wide Web, Florence, Italy, May 2015.

[40] E. Loper and S. Bird, "NLTK: the natural language toolkit", Proceedings of the ACL-02 Workshop on Effective tools and methodologies for teaching natural language processing and computational linguistics, Philadelphia, USA, July 2002.

[41] M. Abadi, et al. "TensorFlow: large-scale machine learning on heterogeneous systems", 0https://www.tensorflow.org/, 2015.

[42] F. Pedregosa, et al. "Scikit-learn: machine learning in python", The Journal of Machine Learning Research, Volume 12, Pages 2825-2830, 2011.

[43] 0https://www.westlaw.com/, Thomson Reuters Corporation, 1975.

[44] 0www.ieee.org, Institute of Electrical and Electronic Engineers, 1963.

[45] 0https://www.elsevier.com/, Elsevier Publishing Group, 1880.

[46] 0https://www.nature.com/, Nature Publishing Group, 1869.

[47] Y. Bengio, A. Courville and P. Vincent, "Representation learning", IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume 35, Issue 8, Pages 1798-1828, August 2013.

[48] F. Hill, K. Cho and A. Korhonen, "Learning distributed representations of sentences from unlabelled data", Volume: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies , San Diego, USA, June 2016.

[49] J. Le and T. Baldwin, "An empirical evaluation of doc2vec with practical insights into document embedding generation", Volume: Proceedings of the 1st Workshop on Representation Learning for NLP, Berlin, Germany, August 2016.

[50] H, Zellig, "Distributional Structure"", Word, Vol. 10, page 146-62, 1954.

[51] C. F. Tsai, "Bag-of-words representation in image annotation: a review", Artificial Intelligence, Volume 2012, 2012.

[52] M. Dammak, M. Mejdoub and C. Ben Amar, "A survey of extended methods to the bag of visual words for image categorization and retrieval", International Conference on Computer Vision Theory and Applications, Lisbon, Protugal, January 2014.

[53] J. Liu, "Image retrieval based on bag-of-words model", https://arxiv.org/abs/1304.51680, 2013.

[54] P. Tan, M. Steinbach and V. Kumar, "Introduction to Data Mining", Addison-Wesley Longman Publishing Co., 2005.

[55] W. Zhang, T. Yoshida and X. Tang, "A comparative study of tf-idf, LSI and multi-words for text classification", Expert Systems with Applications, Vol. 38, No. 3, Pages 2758–2765, 2011.

[56] M. J. Pazzani and D. Billsus,"Content-based recommendation systems", The Adaptive Web, Pages 325–341, Vol. 4321, 2007.

[57] C. Huang, J. Yin and F. HOU, "A text similarity measurement combining word semantic information with TF-IDF method", Chinese Journal of Computers, No. 5, Pages 856-864, 2011.

[58] N. Slonim, N. Friedman and N. Tishby, "Unsupervised document classification using sequential information maximization", Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Tampere, Finland, 2002.

[59] B. Mitra, F. Diaz and N. Craswell, "Learning to match using local and distributed representations of text for web search", Proceedings of the 26th International Conference on World Wide Web, Perth, Australia, 2017.

[60] X. Zhang, J. Zhao and Y. LeCun, "Character-level convolutional networks for text classification", Advances in Neural Information Processing Systems, Montreal, Canada, 2015.

[61] T. K. Landauer, P. W. Foltz and D. Laham, "An introduction to latent semantic analysis", Discourse Processes, Vol. 25, Pages 259-284, 1998.

[62] Y. Gong and X. Liu, "Creating generic text summaries", Sixth International Conference on Document Analysis and Recognition, Sydney, Australia, 2001.

[63] R. Bradford, "Efficient discovery of new information in large text databases", IEEE International Conference on Intelligence and Security Informatics, Atlanta, Georgia, 2005.

[64] P. W. Foltz and S. T. Dumais, "Personalized information delivery: an analysis of information filtering methods", Communications of the ACM, Vol. 34, Pages 51–60, 1992.

[65] J. D. Mcauliffe and D. M. Blei, "Supervised topic models", Advances in Neural Information Processing Systems, Vancouver, Canada, 2008.

[66] M. Hoffman, F. R. Bach and D. M. Blei, "Online Learning for Latent Dirichlet Allocation", Advances in Neural Information Processing Systems, Vancouver, Canada, 2010.

[67] X. Wei and W. B. Croft, "LDA-based document models for ad-hoc retrieval", Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, New York, USA, 2006.

[68] T. Mikolov, M. Karafit, L. Burget, J. Cernock and S. Khudanpur, "Recurrent neural network based language model", INTERSPEECH, Pages 1045–1048, 2010.

[69] T. Kenter and M. de Rijke, "Short text similarity with word embeddings", Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, Melbourne, Australia, 2015.

[70] L. Ma and Y. Zhang, "Using Word2Vec to process big text data", IEEE International Conference on Big Data (Big Data), Santa Clara, USA, 2015.

[71] N. Mrksic, et al., "Counter-fitting word vectors to linguistic constraints", Proceedings of NAACL-HLT, San Diego, USA, 2016.

[72] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengi, and P. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion", The Journal of Machine Learning Research, Vol. 11, Pages 3371-3408, 2010.

[73] K. S. Tai, R. Socher and C. D. Manning, "Improved semantic representations from tree-structured long dhort-term memory networks", Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics, Beijing, China, 2015.

[74] J. Xie, R. Girshick and A. Farhadi, "Unsupervised deep embedding for clustering analysis", Proceedings of the 33rd International Conference on International Conference on Machine Learning, New York, USA, 2016.

[75] M. Chen, "Efficient vector representation for documents through corruption", Proceedings of 5th International Conference on Learning Representations, Toulon, France, 2017.

[76] Sugathadasa, et al., "Legal document retrieval using document vector embeddings and deep learning", Proceedings of Computing Conference, London, UK, 2018.

[77] M. Ranzato and M. Szummer, "Semi-supervised learning of compact document representations with deep networks", Proceedings of the 25th International Conference on Machine Learning, Pages 792–799, New York, USA, 2008.

[78] J. Zheng, Y. Guo, C. Feng and H. Chen, "A hierarchical neural-network-based document representation approach for text classification", Mathematical Problems in Engineering, Vol. 2018, 2018.

[79] J. Tang, et al., "LINE: Large-scale information network embedding", Proceedings of the 24th International Conference on World Wide Web, Pages 1067–1077, Geneva, Switzerland, 2015.

[80] B. Perozzi, R. Al-Rfou and S. Skiena, "DeepWalk: online learning of social representations", Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Pages 701–710, New York, USA, 2014.

[81] k. Grzegorczyk and M. Kurdziel, "Disambiguated skip-gram model", Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Pages 1445–1454, Brussels, Belgium, 2018.

[82] S. Reddy, "Image segmentation by using linear spectral clustering", Journal of Telecommunications System & Management, Vol. 5, No. 3, 2016.

[83] Y. Wu, Y. Bian, X. Zhang, "Remember where you came from: on the second-order random walk based proximity measures", Proceedings of the VLDB Endowment, Pages 13-24, USA, 2016.

[84] W. Xu, X. K and Y. Gong, "Document clustering based on non-negative matrix factorization", Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval, Toronto, Canada, 2003.

[85] H. Cai,V. W. Zheng, K. Chen and C. Chang, "A comprehensive survey of graph embedding: problems, techniques, and applications", IEEE Transactions on Knowledge and Data Engineering, Vol. 30, No. 9, Pages 1616–1637, 2018.

[86] D. Yang, Z. Ma and A. Buja, "A sparse SVD method for high-dimensional data", Journal of Computational and Graphical Statistics, Vol. 23, Pages 923-942, 2014.

[87] X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques", Advances in Artificial Intelligence, Vol. 2009, January 2009.

[88] Yang, et al., "Collaborative filtering-based recommendation of online social voting", IEEE Transactions on Computational Social Systems, Vol. 4, No. 1, 2017.

[89] R. Sharma, D. Gopalani and Y. Meena, "Collaborative filtering-based recommender system: Approaches and research challenges", 3rd International Conference on Computational Intelligence Communication Technology (CICT), India, 2017.

[90] C. Wei, C. Yang and H. Hsiao, "A collaborative filtering-based approach to personalized document clustering", Decision Support Systems, Vol. 45, No. 3, Pages 413–428, 2008.

[91] L. Sheng, J. Kawale and Y. Fu, "Deep collaborative filtering via marginalized denoising auto-encoder", Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, Melbourne, Australia, 2015.

[92] R. Salakhutdinov, A. Mnih and G. Hinton, "Restricted Boltzmann machines for collaborative filtering", Proceedings of the 24th International Conference on Machine Learning, Corvalis, Oregon, USA, 2007.

[93] Zhao, et al. "A heuristic approach to determine an appropriate number of topics in topic modeling", BMC Bioinformatics, Vol. 16, 2015.

[94] Y. W. Teh, M. I. Jordan, M. J. Beal and D. M. Blei, "Hierarchical Dirichlet processes", Journal of the American Statistical Association, Vol. 101, Pages 1566–1581, 2006.

[95] Y. Bengio, A. Courville and P. Vincent, "Representation Learning: A Review and New Perspectives",IEEE Transactions on Software Engineering, Pages 1798-1828, August 2013.

[96] R. Baeza-Yates and B. Ribeiro-Neto, "Modern information retrieval: the concepts and technology behind search", Addison-Wesley Publishing Company, 2008.

[97] H. Cai, V. W. Zheng and C. Chang, "A comprehensive survey of graph embedding: problems, techniques and applications", IEEE Transactions on Knowldege and Data Engineering, Vol 30, No. 9, 2018.

[98] B.ZahirAzami, V. Tabatabaee, B. AzimiSadjadi and H. Borhani, "Encoder Decoder Neural Network", International Conference on Neural Networks Applications on Signal Processing ICNNASP, Singapore, August 1993.