Discovery and Interpretation of Embedding Models for Knowledge Representation

Thesis submitted in partial fulfillment of the requirements for the degree of

MS by Research in Computer Science and Engineering

by

Ganesh J
201407652
ganesh.j@research.iiit.ac.in

International Institute of Information Technology
Hyderabad - 500 032, INDIA
June, 2017
International Institute of Information Technology  
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled “Discovery and Interpretation of Embedding Models for Knowledge Representation” by Ganesh J, has been carried out under our supervision and is not submitted elsewhere for a degree.

_17.06.2017_  
Date

Advisor: Prof. Vasudeva Varma

_17 June 2017_  
Date

Advisor: Prof. Manish Gupta
To my dearest family and friends
for their endless love and support.
Acknowledgments

I would like to take this opportunity to express my heartfelt gratitude towards everyone who not only helped me in completing this thesis but also made my experience at IIIT Hyderabad, a memorable, enjoyable and productive one.

This thesis would not be possible without the guidance of Prof. Vasudeva Varma. He gave me complete autonomy in choosing my research problem, which significantly helped me in honing my research skills. He always encouraged me to think big with respect to the problem at hand. Despite his busy schedule, he has meticulously read my paper and provided valuable insights in writing a gripping paper. I owe a big salute to him for his course, ‘Information Retrieval and Extraction’ offered in the Spring 2015. The course was set to be light, sometimes deliberately vague in explaining the fundamentals of Information Retrieval, which allowed me to think a lot in the class hours. Eventually this course laid out the foundation for my research.

I am indebted to my co-advisor, Prof. Manish Gupta, who is incredibly knowledgeable, and I feel he has really gone out of his way to mentor me. His ideas have always helped me in deepening my understanding of the problem. His thorough review has helped me in writing and positioning the paper nicely. Thanks to him, I have understood the importance of doing hard work, picking up worthwhile problems, presenting the concepts and managing the time well.

I am thankful to my professors at IIIT Hyderabad who helped me in building strong fundamentals in subjects such as Statistical Methods in A. I., Optimization Methods and Data Warehousing and Mining. I am thankful to agencies such as Microsoft Research, Special Interest Group on Information Retrieval and Neural Information Processing Systems Foundation for sponsoring my travel to attend international conferences. In addition, I would like to thank Mahender for being always reachable for handling all the logistics.

Without the continued support and sacrifices from my family, I would not have completed my masters on time. Particularly, I extend my gratitude to my mother for letting me carve my own path. I am lucky to have my best friends, Vijay and Yasaswi for making my stay at Hyderabad pleasant and memorable. In addition I would like to thank both the present and the past M. S. as well as PhD students, whose interactions helped me a lot. Finally, I would like to thank the Almighty and a special person, who helped me in understanding my potential, giving a pat on the back during tough times and ceaselessly loving me.
Abstract

Recently representation learning has been receiving much attention from both industrial and academic communities. Thanks to the advent of powerful GPUs and advanced techniques like dropout and rectified linear units, the interest in neural networks has been revived. The current state-of-the-art solution for the applications in the field of Natural Language Processing (NLP) and Information Retrieval (IR) is powered by neural network based representation learning models. In this thesis, we focus on the unsupervised representation learning models which are cheap to build (as they rely on unlabeled data) but very effective for many downstream applications. We explore two main challenges in building an unsupervised representation learning model for NLP and IR problems.

Context serves as the source of knowledge for estimating the representation of data. For example, in Word2Vec, the context is the set of words surrounding a given word in a sentence. The first challenge we focus is the context insufficiency problem. For instance, consider the models used in practise to generate representations for tweets. We observe that tweets do not exist in isolation, and hence the performance of the models which work only with the content of the tweet (as context) is found to be sub-optimal. To handle this issue, we propose a better model which also captures the interactions between the tweet and its adjacent tweet (as context) in the users’ timeline. Along similar lines, we explore two more use cases where we smartly incorporate novel contexts that captures complex interactions such as scientific author and his/her paper interaction in an author collaboration network (‘Author2Vec’) and sentences interaction in a document (‘Doc2Sent2Vec’), that turned out to be advantageous in computing accurate author and document representations respectively. We conclude that if we smartly leverage the available contexts, the performance of the model improves significantly.

Though the representation learning models perform well in practise, little is known about the core properties of the data encoded within the representations. Understanding these core properties would empower us in making generalizable conclusions about the quality of the representations. Hence, the second challenge we focus is the human interpretability problem with these automatically learned representations. For instance, researchers in Twitter analytics are getting interesting results by applying different representation learning models for several valuable tasks such as sentiment analysis, semantic textual similarity computation, microblog retrieval, hashtag identification and so on. In order to understand the core properties encoded in a tweet representation, we evaluate the representations to estimate the extent to which it can model each of those properties such as tweet length, presence of words, hashtags, mentions, capitalization and so on. This is done with the help of multiple classifiers which take
the representation as input. Essentially, each classifier evaluates one of the syntactic or social properties which are arguably salient for a tweet. The result is an application independent, fine-grained analysis of tweet representations generated by different representation learning models.

This thesis is one of the initial work to overcome the above-mentioned challenges by proposing novel methods to improve the contexts and interpret the representations. In the growing body of representation learning research, the set of models and framework proposed in this thesis could act as the basic building blocks in the future works attempting to advance the science of building smarter NLP/IR systems.
## Contents

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td><strong>Introduction</strong></td>
<td>1</td>
</tr>
<tr>
<td>1.1 Motivation</td>
<td>2</td>
</tr>
<tr>
<td>1.1.1 Building Sentence / Document Representation Models</td>
<td>2</td>
</tr>
<tr>
<td>1.1.1.1 Applications</td>
<td>2</td>
</tr>
<tr>
<td>1.1.1.2 Challenges</td>
<td>2</td>
</tr>
<tr>
<td>1.1.2 Building Node Representation Models</td>
<td>3</td>
</tr>
<tr>
<td>1.1.2.1 Applications</td>
<td>3</td>
</tr>
<tr>
<td>1.1.2.2 Challenges</td>
<td>3</td>
</tr>
<tr>
<td>1.1.3 Understanding Sentence Representation Models</td>
<td>4</td>
</tr>
<tr>
<td>1.2 Summary of Thesis</td>
<td>4</td>
</tr>
<tr>
<td>1.2.1 Problem Statement</td>
<td>4</td>
</tr>
<tr>
<td>1.2.1.1 Improving Semantic Representations through Novel Context</td>
<td>4</td>
</tr>
<tr>
<td>1.2.1.2 Interpreting Semantic Representations</td>
<td>5</td>
</tr>
<tr>
<td>1.2.2 Contribution</td>
<td>5</td>
</tr>
<tr>
<td>1.2.3 Organization of the Thesis</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td><strong>Summary of Existing Embedding Models and Related Works</strong></td>
<td>8</td>
</tr>
<tr>
<td>2.1 Existing Sentence / Document Embedding Models</td>
<td>8</td>
</tr>
<tr>
<td>2.2 Existing Node Embedding Models</td>
<td>10</td>
</tr>
<tr>
<td>2.3 Understanding Sentence Embedding Models</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>11</td>
</tr>
<tr>
<td><strong>Doc2Sent2Vec - A Novel Two-Phase Approach for Learning Document Representation</strong></td>
<td>11</td>
</tr>
<tr>
<td>3.1 Overview</td>
<td>11</td>
</tr>
<tr>
<td>3.2 Problem Statement</td>
<td>12</td>
</tr>
<tr>
<td>3.3 Proposed Approach</td>
<td>12</td>
</tr>
<tr>
<td>3.3.1 Modeling word-level coherence</td>
<td>12</td>
</tr>
<tr>
<td>3.3.2 Modeling sentence-level coherence</td>
<td>14</td>
</tr>
<tr>
<td>3.3.3 Training details</td>
<td>15</td>
</tr>
<tr>
<td>3.4 Experiments and Analysis</td>
<td>15</td>
</tr>
<tr>
<td>3.4.1 Dataset Description</td>
<td>15</td>
</tr>
<tr>
<td>3.4.2 Experimental Setup</td>
<td>16</td>
</tr>
<tr>
<td>3.4.3 Analysis on CND</td>
<td>17</td>
</tr>
<tr>
<td>3.4.4 Analysis on Wiki10+</td>
<td>17</td>
</tr>
<tr>
<td>3.5 Summary</td>
<td>17</td>
</tr>
</tbody>
</table>
## CONTENTS

4  Author2Vec - Learning Author Representations by Combining Content and Link Information 19
   4.1 Overview ................................. 19
   4.2 Problem Statement .............................. 19
   4.3 Proposed Approach ............................ 20
      4.3.1 Content-Info Model ....................... 20
      4.3.2 Link-Info Model ......................... 21
      4.3.3 Training Details ......................... 21
   4.4 Experiments and Analysis .................. 21
      4.4.1 Task Description ....................... 22
      4.4.2 Result Analysis ......................... 22
   4.5 Summary ................................ 23

5  Improving Tweet Representations using Temporal and User Context ......................... 24
   5.1 Overview ................................ 24
   5.2 Problem Statement ......................... 24
   5.3 Proposed Approach ........................... 25
      5.3.1 Word Context Model ..................... 26
      5.3.2 User + Tweet Context Model .......... 26
      5.3.3 Training Details ....................... 27
   5.4 Experiments and Analysis .................. 27
      5.4.1 Task Description ....................... 27
      5.4.2 Experimental Setup ..................... 28
      5.4.3 Comparative Analysis .................. 28
      5.4.4 Impact of Variable Attention .......... 29
   5.5 Summary ................................ 30

6  Interpretation of Semantic Tweet Representations .................................. 31
   6.1 Overview ................................ 31
   6.2 Elementary Property Prediction Tasks ..................................... 31
      6.2.1 Syntactic Tasks ......................... 32
      6.2.2 Social Tasks ............................. 33
   6.3 Tweet Embedding Models .................................. 34
      6.3.1 Unsupervised ............................ 34
      6.3.2 Supervised .............................. 35
   6.4 Experiments ................................ 35
   6.5 Results and Analysis .......................... 37
      6.5.1 Property Prediction Task Accuracy .... 38
      6.5.2 Sensitivity to Tweet Length .......... 38
      6.5.3 Embedding Norm vs Tweet Length .......... 39
      6.5.4 Representation Size vs Property Prediction Task Accuracy ......... 39
      6.5.5 Sensitivity of Property prediction task to Word Order ........... 41
      6.5.6 A comment on BOW vs Paragraph2Vec .......... 41
   6.6 Summary ................................ 41

7  Conclusions and Future Work ........................................ 43
   7.1 Conclusion ................................ 43
   7.2 Future Work ................................ 43
CONTENTS

Bibliography ........................................... 46
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Architecture diagram of the Doc2Sent2Vec approach. Sentence embedding weights are shared between the two tiers.</td>
<td>12</td>
</tr>
<tr>
<td>5.1</td>
<td>Temporal Context with Variable Attention Example - ( t(j - 1), t(j - 2), t(j + 1) ) and ( t(j + 2) ) form the temporal context of ( t(j) ). ( \alpha )'s denote the attention parameters of the proposed model.</td>
<td>25</td>
</tr>
<tr>
<td>5.2</td>
<td>Architecture diagram of our model.</td>
<td>26</td>
</tr>
<tr>
<td>5.3</td>
<td>Model performance w.r.t. temporal context size ( C_T ).</td>
<td>28</td>
</tr>
<tr>
<td>5.4</td>
<td>Mean attention w.r.t. distance from the center ( C_T ).</td>
<td>29</td>
</tr>
<tr>
<td>5.5</td>
<td>Mean attention w.r.t. epoch for our model when the user context is included in the temporal context.</td>
<td>29</td>
</tr>
<tr>
<td>6.1</td>
<td>Sensitivity of the models to Tweet Length</td>
<td>37</td>
</tr>
<tr>
<td>6.2</td>
<td>Embedding Norm versus Tweet Length - Only for unsupervised models</td>
<td>39</td>
</tr>
<tr>
<td>6.3</td>
<td>Model Performance for varying representation size and word order - Only for supervised models. The suffix ‘P’ indicates that the model is built and tested on tweets whose words are randomly ordered.</td>
<td>40</td>
</tr>
</tbody>
</table>
# List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1 Dataset Details</td>
<td>14</td>
</tr>
<tr>
<td>3.2 Classification Performance on CND Dataset</td>
<td>16</td>
</tr>
<tr>
<td>3.3 Classification Performance on Wiki10+ Dataset</td>
<td>17</td>
</tr>
<tr>
<td>4.1 Performance comparison on link prediction and clustering tasks</td>
<td>22</td>
</tr>
<tr>
<td>5.1 User profile attribute classification - F1 Score</td>
<td>28</td>
</tr>
<tr>
<td>6.1 Elementary Property Prediction Task F1-Score (%) - Performance Comparison</td>
<td>36</td>
</tr>
<tr>
<td>6.2 BOW versus Paragraph2Vec - Performance Comparison (F1-Score (%)) on multiple tasks</td>
<td>41</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 Motivation

Most of the machine learning algorithms such as Support Vector Machines (SVM), Neural Networks (NN) require the input data to be represented as a fixed-length feature vector. Understanding how to best capture the essential meaning of a data in a machine-understandable format (or “representation”) is central to the performance of several downstream applications [26, 27] such as sentiment analysis of documents, document retrieval, user profiling, user link prediction, user visualization and so on. Recently, there has been a paradigm shift in machine learning towards using distributed representations (aka ‘embeddings’\(^1\)) for words [6, 18, 11], sentences [30, 32], documents [3, 4, 7] and nodes in a graph [16]. Though these automatically learned representations are hard to understand for a human, they have the following advantages: (1) they work well in practice, (2) they reduce manual efforts, which sometimes introduce errors, and (3) they reduce the dependence on domain level experts. These representation learning methods have proven to be very effective not only for several Natural Language Processing (NLP) tasks [28] like statistical machine translation, sentiment classification, question answering, paraphrase identification, modeling text interestingness, but also for graph mining tasks such as link prediction, node classification, recommendation and visualization.

We posit that there are two main challenges in the current representation learning models. The first challenge is the problem of context insufficiency i.e., under utilization of the available context in a given data. For instance, in Twitter, we observe modeling a tweet using the content of the tweet to be a sub-optimal strategy leading to poor performance of the model in downstream applications. In this thesis, we argue that exploiting available contexts (like surrounding tweets in users’ timeline) smartly could enhance the performance of the model by a large margin. To this end, we validate this claim by building accurate document, author and tweet representations for three application domains: documents, author-collaboration network and tweets respectively, through novel contexts. The second challenge is the interpretability problem with these automatically learned representations. Although the researchers are getting interesting results by applying different representation learning models for several valuable tasks

\(^1\)Note that the word ‘embedding’ and ‘representation’ mean the same and they would be used interchangeably in the thesis.
in NLP and IR, little is known about the core properties encoded by the representations generated from these models, knowing which will allow us to make generalizable conclusions. To this end, we present the first step in opening the black-box of vector embeddings for Twitter posts. In the next section, we will highlight the applications and challenges of building document, node representation models and in understanding the sentence representation models.

### 1.1.1 Building Sentence / Document Representation Models

In this section we will firstly understand some of the applications of building document representation models. Then we will present the challenges in building such models.

#### 1.1.1.1 Applications

- **Topic Identification** - Given a document, build a model like SVM which takes the document representation as input and predict the topic of the document. The number of topics for a document can be more than one.

- **Document Visualization** - Identify the natural document clusters using clustering algorithms like k-means given the document representation for a set of documents.

- **Document Similarity Computation** - Given two documents, build a model like SVM which takes the representation for both the documents to compute the semantic similarity of them.

- **Document Retrieval** - Given a query, build a model like NN that rank documents using their representations according to their relevancy.

#### 1.1.1.2 Challenges

- **Data Sparsity and High Dimensionality** - Most of the ML models suffer from the curse of dimensionality and hence cannot work well with sparse and high-dimensional document representation. Bag-Of-Words (BOW) [24] is the most well-known document representation method and suffers from these problems. Although the neural representation learning models overcomes the dimensionality issue, there is a need to develop smart techniques to avoid sparsity issue. In this thesis, we frame this problem as a context insufficiency problem in neural models and propose novel contexts to overcome it.

- **Efficiency** - The running time of the traditional models such as LSA and LDA [25] grows with the size of the corpora. The recent neural representation learning models are known to be efficient as they are mostly scalable and parallelizable.
• **Effectiveness** - The representation generated by the models must be helpful for one or more downstream applications. The recent neural representation learning models have proven to be effective and sometimes applied to text data from different domains, with minimal changes.

The above-mentioned applications and challenges apply when building sentence or tweet representation models as well.

### 1.1.2 Building Node Representation Models

Recently, there has been an increasing interest in embedding information networks [15, 16] into low-dimensional vector spaces. The motivation is that once the embedded vector form is obtained, the network mining tasks can be solved by off-the-shelf machine learning algorithms. In an attempt to construct good representation in a scalable way, researchers have started using deep learning as a tool to analyze graphs. In this section we will firstly understand some of the applications of building node representation models. Then we will present the challenges in building such models.

#### 1.1.2.1 Applications

- **Node Classification** - Given a node in the graph, build a model like SVM which takes the node representation as input and predict the class the node belongs to. The number of classes for a node can be more than one.

- **Link Prediction** - Given two nodes in the graph, build a model like SVM which takes the representation for both the nodes to predict whether there exist a edge connecting them.

- **Node Recommendation** - Recommend a node for a given node in the graph.

- **Node Visualization** - Identify the natural node clusters using clustering algorithms like k-means given the node representation for a set of nodes in the graphs.

#### 1.1.2.2 Challenges

- **Link Sparsity** - The real world information network has much more links than that is observed [19]. We find the models which relies on only the observed links tending to be sub-optimal for downstream applications. In this thesis, we frame this problem as a context insufficiency problem in neural models and propose novel context to overcome it.

- **Efficiency** - The running time of the traditional models which relies on matrix factorization techniques grows with the size of the corpora. The recent neural representation learning models are known to be efficient as they are mostly scalable and parallelizable.
• **Effectiveness** - The representation generated by the models must be helpful for one or more downstream applications. The recent neural representation learning models have proven to be effective and sometimes applied to different graph data, with minimal changes.

1.1.3 **Understanding Sentence Representation Models**

There is a huge potential if we can create techniques to interpret the representations generated from the representation learning models. Since these representations are not very human interpretable, such techniques help us in unearthing the basic characteristics of different representations. This surely allow us to make generalizable conclusions. In this thesis, we present a strategy to accomplish the task of understanding the tweet representations.

1.2 **Summary of Thesis**

In this section we provide a brief overview of the thesis. We introduce the problem statement and list the contributions made in the thesis followed by the organization of the thesis.

1.2.1 **Problem Statement**

In this thesis, we propose techniques to improve and interpret the semantic representations obtained by neural representation learning models for NLP and IR applications.

1.2.1.1 **Improving Semantic Representations through Novel Context**

We discover that the effective use of available context is vital for computing accurate semantic representations to alleviate the impact of data sparsity. To this end, we will list down the research areas touched upon by this thesis.

- **Document Representation** - We propose an approach, Doc2Sent2Vec which estimates the document representation accurately by utilizing both the word-level context (set of words surrounding the given word in the sentence while modeling the sentence) and sentence-level context (set of sentences surrounding the given sentence in the document while modeling the document). The main motivation to propose the sentence-level context is the empirical observation that ParagraphVec [3] which relies only on the word-level context performs poorly for scientific article and Wikipedia page classification tasks, when compared to Doc2Sent2Vec.

- **Node Representation** - We propose a novel model, Author2Vec which computes the representation for a scientific author from bibliographic co-authorship networks such that authors who work in similar research area comes closer in the vector space, by utilizing both the content context (set of paper written by the given author) and link context (set of authors collaborated with the given
author). The main motivation to propose the content context is to overcome the link sparsity problem [19] faced by DeepWalk [16] which relies only on the link context. For example, two authors who write scientific articles related to the field ‘Machine Learning’ are not considered to be similar by DeepWalk if they are not connected. Author2Vec overcomes the above mentioned problem by fusing the textual information with the link information in a synergistic fashion, for creating author representations, and outperforms DeepWalk for link prediction and clustering tasks.

- **Tweet Representation** - We propose a novel model which estimates the tweet representation accurately by utilizing multiple contexts such as the word-level context (set of words surrounding the given word in the tweet while modeling the tweet), temporal context (set of tweets surrounding the given tweet in the Twitter users’ timeline while modeling the tweet) and user context (set of tweets written by the given user). The main motivation to propose the temporal and user context is the empirical observation that ParagraphVec [3] which relies only on the word-level context performs poorly for entity classification task, when compared to our model which smartly utilizes the useful information flowing from the surrounding tweets and the set of posts written by the user.

### 1.2.1.2 Interpreting Semantic Representations

In this thesis, we focus only on interpreting the tweet representations. In order to understand the core properties encoded in a tweet representation, we evaluate the representations to estimate the extent to which it can model each of those properties such as tweet length, presence of words, hashtags, mentions, capitalization, and so on. This is done with the help of multiple classifiers which take the representation as input. Essentially, each classifier evaluates one of the syntactic or social properties which are arguably salient for a tweet. We assume that if we cannot train a classifier to predict a property based on its tweet representation, then this property is not encoded in this representation. For example, the model which preserves the tweet length should perform well in predicting the length given the representation generated from the model. Though these elementary property prediction tasks are not directly related to any downstream application, knowing that the model is good at modeling a particular property (e.g., the social properties) indicates that it could excel in correlated applications (e.g., user profiling task). In this work we perform an extensive evaluation of 9 unsupervised and 4 supervised tweet representation models, using 13 different properties.

### 1.2.2 Contribution

We make the following contributions in the thesis:

**Doc2Sent2Vec: A Novel Two-Phase Approach for Learning Document Representation**
• We present a novel two-phase approach, Doc2Sent2Vec, to learn document embeddings. To this end, we introduce a novel sentence-level language model which effectively constructs document representations by explicitly modeling sentence-level coherence.

• Experiments on Citation Network and Wikipedia datasets show that Doc2Sent2Vec learns high quality document embeddings outperforming the competitive baselines in both the classification tasks.

Author2Vec: Learning Author Representations by Combining Content and Link Information

• We propose a novel model, Author2Vec which fuses content and link information to learn high quality author embeddings given a bibliographic network.

• We show the efficacy of our author representations over DeepWalk for two tasks: link prediction and clustering.

Improving Tweet Representations using Temporal and User Context

• Our work is the first to model the semantics of the tweet using the temporal context. We hypothesize that they are a rich carrier of semantics with respect to the tweet being modeled.

• We introduce a novel attention based model that learns the weights for tweets in the context by back-propagating semantic loss. This setup helps the model assign appropriate weights proportional to the semantic relevance of context tweets with respect to the tweet being modeled.

• We propose a novel way to learn user vector summarizing the content the user writes, which in turn helps in enriching the quality of the tweet embeddings.

• We conduct quantitative analysis to showcase the application potential of the tweet representations learned from the model and also provide some interesting findings.

Interpretation of Semantic Tweet Representations

• Our work is the first towards interpreting the tweet embeddings in a fine-grained fashion. To this end, we propose a set of tweet-specific elementary property prediction tasks which help in unearthing the basic characteristics of different tweet representations.

• To the best of our knowledge, this work is the first to do a holistic study of traditional, unsupervised and supervised representation learning models for tweets.

• We compare various tweet representations with respect to such properties across various dimensions like tweet length, embedding norm, representation size and word ordering sensitivity.
1.2.3 Organization of the Thesis

The thesis is organized as follows. In Chapter 2 we not only review the various representation learning models proposed for entities such as document, graph and tweets, but also briefly see the approaches used so far in understanding sentence representation learning models. In Chapter 3 we introduce the Doc2Sent2Vec approach and present its theory and efficacy. In Chapter 4 we introduce the Author2Vec model and present its theory and efficacy. In Chapter 5 we introduce the novel tweet representation model and present its theory and efficacy. In Chapter 6 we present our approach in interpreting the tweet representations along with some interesting results. We finally conclude the thesis and discuss future directions in Chapter 7.

In Section 1.1, we understand the application potential and challenges in building and interpreting representation models. We then summarized the problem statement and the contributions along with the thesis organization in Section 1.2. We discuss the relevant work done in the field of representation learning for NLP and IR area in the next chapter.
Chapter 2

Summary of Existing Embedding Models and Related Works

In this chapter we will discuss the existing work done in building sentence / document and node embedding models. We also briefly present the works carried out in understanding the sentence embedding models.

2.1 Existing Sentence / Document Embedding Models

In this section we list some of the unsupervised and supervised sentence / document embedding models from the literature.

(a) **Deep Structured Semantic Model (DSSM)**: DSSM [37] uses a deep feed forward network to represent text strings in a continuous semantic space and modeling the semantic similarity between two text strings. Specifically, this model is trained by maximizing the conditional likelihood of the clicked documents given a query using the clickthrough data.

(b) **Convolutional Deep Structured Semantic Model (CDSSM)**: CDSSM [38] is a convolutional variant of DSSM capturing both the word n-gram level and sentence-level contextual structures for IR using carefully designed convolution and pooling operations.

(c) **Paragraph2Vec**: Paragraph2Vec [3] extends the Word2Vec network to learn an embedding for document which are good in predicting the words within the document. This model has been shown to outperform several sophisticated models in applications such as sentiment analysis and document retrieval. Dai et al. [4] allows the gradient to update the word vectors in the Paragraph2Vec network and present interesting results in Wikipedia browsing and arXiv browsing.

(e) **Sentiment Specific Word Embedding (SSWE)**: SSWE [29] is a simple feed forward network that learns sentiment aware word embeddings using distant supervision (emoticons). This model improves the state-of-the-art in sentiment analysis for tweets.

(d) **Hierarchical Document Vector (HDV)**: HDV [7] extended Paragraph2Vec model by learning document embedding which are good in predicting the words within it and also the documents surrounding it in a user stream. This model outperforms Paragraph2Vec in downstream applications such as movie genre classification and large-scale document representation.
(e) **Skip-Thought vectors**: Skip-Thought vectors [34] uses a Gated Recurrent Unit (GRU) [41] based Encoder-Decoder network to learn generic sentence embedding which are good in predicting the surrounding sentences (sentential context). This model outperformed the state-of-the-art model in tasks such as semantic relatedness computation, paraphrase detection, image-sentence ranking and sentence classification.

(f) **Sequential Denoising Autoencoders (SDAE)**: SDAE [32] uses a Long-Short Term Memory (LSTM) [40] Encoder-Decoder network which predicts the source sentence given the corrupted version of the source sentence. The trained encoder proves to be effective in several tasks such as semantic relatedness computation and sentence classification.

(f) **FastSent**: FastSent [32] works with the same objective as Skip-Thought vectors using a Word2Vec architecture. The simplicity of the architecture reduced the running time of the model significantly without compromising on the performance in the downstream applications.

(g) **Siamese CBOW**: Siamese CBOW [30] works with the same objective as Skip-Thought vectors using a Siamese architecture. The resulting representations improved the state-of-the-art in semantic relatedness computation.

(h) **Tweet2Vec**: Tweet2Vec [31] uses a Bi-GRU Encoder network which learns tweet embedding directly from characters using hashtags for supervision. The resulting representations improved the state-of-the-art in hashtag prediction in Twitter. Vosoughi et al. [42] propose a CNN-LSTM Encoder-Decoder network that predicts the tweet itself. They also use data augmentation techniques for regularization and shows the superiority of the model in applications such as semantic relatedness computation and sentiment classification.

(i) **Character to Sentence Convolutional Neural Network (CharSCNN)**: CharSCNN [46] is a novel sentiment classification model using concatenation of word vectors and CNN on top of character vectors. This model is capable of exploiting important information that appears in different parts of a word.

(j) **Tree-LSTM**: Tree-LSTM [20] is a novel recursive neural network that generalizes LSTM to model recursive nature of sentences. This model outperforms the state-of-the-art models in computing semantic relatedness and sentiment classification.

(k) **FastText**: FastText [36] is a simple feed forward network that classify any text strings using the average of word vectors. This models performs competitively with several sophisticated models in downstream applications such as sentiment analysis and tag prediction.

(l) **MemNet**: MemNet [43] is an extension of Memory Networks [49] to classify the sentiment of an aspect using Deep Memory Networks. The resulting representations improved the state-of-the-art in aspect based sentiment analysis.

Models from (a) to (h) fall in the unsupervised category, while the models from (i) to (l) fall in the supervised category. Based on the network architecture, all the models can be classified into one or more of the following categories: Feed Forward, Word2Vec, Encoder-Decoder, Siamese, Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), Recursive Neural Network (ReNN) and Memory Networks (MemNN). Note that recursive neural networks work with the parse tree, and hence
are ill-suited for representing tweets, as the parse tree construction is not only computation intensive but also expects the input sentences to be grammatically well-formed unlike most tweets.

2.2 Existing Node Embedding Models

Deep learning allows the researchers to analyze the graphs by constructing good representation in a scalable way. DeepWalk [16] is a recent model that transforms a graph structure into a sample collection of linear sequences containing vertices using uniform sampling (truncated random walk). They treat each sample as a sentence, run the Skip-gram model [18], originally designed for learning word representations from linear sequences to learn the representation of vertices, from such samples. The main drawback of DeepWalk is the link sparsity problem [19] inherent in a real world information network. For example, two authors who write scientific articles related to the field ‘Machine Learning’ are not considered to be similar by DeepWalk if they are not connected.

2.3 Understanding Sentence Embedding Models

In a recent work, Hill et al. [32] perform a comparison of different sentence representation models by evaluating them for different high-level semantic tasks such as paraphrase identification, sentiment classification, question answering, document retrieval and so on. This type of coarse-grained analysis is opaque as it does not clearly reveal the kind of information encoded by the representations. Adi et al. [33] investigates three sentence properties namely sentence length, content and word order in comparing unsupervised sentence representation models such as average of words vectors and LSTM auto-encoders.

In the next chapter, we begin with Doc2Sent2Vec approach which accurately computes the document representation using multiple contexts. This is the first usecase we use to highlight the context insufficiency problem in the representation learning models.
Chapter 3

Doc2Sent2Vec - A Novel Two-Phase Approach for Learning Document Representation

3.1 Overview

Document representations play a vital role in the performance of several downstream IR applications such as document classification (or tagging), retrieval, ranking and so on. The most commonly used document representation is bag-of-words (BOW) or bag-of-n-grams [24]. Despite its simplicity and efficiency, it fails to capture the semantics of the documents as it suffers from data sparsity and curse of high dimensionality. Latent Dirichlet Allocation (LDA) [25] is another widely adopted distributed document representation.

In an attempt to harness the power of neural networks for document representations, Le et al. [3] proposed a simple approach to learn document embedding from the word sequence using a standard word-level language model. The representations learned capture the ordering of words (unlike BOW model) and also the semantics of the words in an efficient way (unlike LDA model). In their follow-up work [4], the authors proposed an incremental model by jointly learning the word embeddings along with its document embedding. This change leads to learning rich and accurate representation compared to the previous model, which freezes the word vectors while learning the document vectors.

Inspired by the superior results obtained by the neural language models, we present a two-phase approach, Doc2Sent2Vec, to learn document embedding. In the first phase, we learn the sentence embedding using the word sequence generated from the sentence. Intuitively, the sentence representation is computed by modeling word-level coherence. In the next phase, we propose a novel model that learns the document representation from the sentence sequence generated from the document. Intuitively, the document embedding is computed by modeling sentence-level coherence. We argue in this work that the proposed decoupled strategy allows our model to compute accurate and rich document representations.

We validate the learned document embeddings using two classification tasks. In the first task, we aim at classifying a research article (or paper) among one of the eight different fields of computer science domain. In the second task, we predict the tag of a Wikipedia page. Doc2Sent2Vec outperforms the
Figure 3.1: Architecture diagram of the Doc2Sent2Vec approach. Sentence embedding weights are shared between the two tiers.

existing state-of-the-art model in scientific article classification task by \( \sim 12.07\% \) and Wikipedia page classification task by \( \sim 6.93\% \), both in terms of \( F_1 \) score.

### 3.2 Problem Statement

Formally, let us denote a set \( D \) of \( M \) documents, \( D = \{d_1, d_2, \cdots, d_M\} \), where each document \( d_m \) is a sequence of \( T_m \) sentences, \( d_m = \{s_{(m,1)}, s_{(m,2)}, \cdots, s_{(m,T_m)}\} \). Each sentence is a sequence of \( T_n \) words, \( s_{(m,n)} = \{w_{(m,n,1)}, w_{(m,n,2)}, \cdots, w_{(m,n,T_n)}\} \). For brevity, we drop the sentence index \( m \) when it is obvious in the context. The goal of the Doc2Sent2Vec approach is to jointly learn low-dimensional representations of words, sentences and documents as a continuous feature vector of dimensionality \( D_w \), \( D_s \) and \( D_d \) respectively. We will realize this goal in two phases, as discussed in the following section.

### 3.3 Proposed Approach

Our hierarchical framework as shown in Figure 3.1 consists of two tiers; one learning the sentence representation from the word context, another learning the document representation from the sentence context.

#### 3.3.1 Modeling word-level coherence

In the first phase, the model aims to learn sentence representation from the word sequence within the sentence. We add the sentence vector to the standard language model that predicts the next word given its context word. This sentence vector must capture the topics of the sentence in a compact form. Each word is mapped to a unique vector, denoted by a column in the matrix \( V_{\text{word}} \), whose size is given by
\(D_w \times |V_w|\) (where \(|V_w|\) is the vocabulary size). Similarly, each sentence in a document is mapped to a unique vector denoted by a column in the matrix \(V_{sent}\) with size \(D_s \times |V_s|\), where \(|V_s|\) is the number of unique sentences. The model uses the concatenation of word vectors of context words along with the sentence vector as features to predict the given word in a sentence.

Formally, consider \(w_{(n,t-c_w)}, \cdots, w_{(n,t-1)}, w_{(n,t+1)}, \cdots, w_{(n,t+c_w)}\) as the context words for the target word \(w_{(n,t)}\), appearing in the sentence \(s_{(m,n)}\). The objective of the word-level language model is to maximize the log likelihood probability.

\[
\mathcal{L}_{\text{word}} = \sum_{d_m \in D} \left[ \sum_{s_{(m,n)} \in d_m} \log \mathbb{P}(s_{(m,n)} | w_{(1)}, \cdots, w_{(n,T)}) + \right. \\
\left. \sum_{s_{(m,n)} \in d_m} \sum_{w_{(n,t)} \in s_{(m,n)}} \log \mathbb{P}(w_{(n,t)} | w_{(n,t-c_w)}, \cdots, w_{(n,t-1)}, w_{(n,t+1)}, \cdots, w_{(n,t+c_w)}; s_{(m,n)}) \right] 
\]

(3.1)

Here \(2 \times c_w\) denotes the length of the context for the word sequence. The probability of observing the central word \(w_{(n,t)}\) given the context words and the sentence is defined using the following softmax function.

\[
\mathbb{P}(w_{(n,t)} | w_{(n,t-c_w)}, \cdots, w_{(n,t-1)}, w_{(n,t+1)}, \cdots, w_{(n,t+c_w)}; s_{(m,n)}) = \frac{\exp(\bar{v}_{\text{word}}^T w_{(n,t)})}{\sum_{w=1}^{|V_w|} \exp(\bar{v}_{\text{word}}^T v_w)}
\]

where \(v_{w_{(n,t)}}\) is the output representation of \(w_{(n,t)}\) and \(\bar{v}_{\text{word}}\) is the concatenation of the input embeddings (ignoring the central term \(w_{(n,t)}\)), with dimensionality \(2 \times c_w \times D_w + D_s\).

\[
\bar{v}_{\text{word}} = [v_{s_{(m,n)}}; v_{w_{(n,t-c_w)}}; \cdots; v_{w_{(n,t-1)}}; v_{w_{(n,t+1)}}; \cdots; v_{w_{(n,t+c_w)}}]
\]

(3.2)

Similarly, we define the probability of observing the sentence \(s_{(m,n)}\), given the words present in it as follows.

\[
\mathbb{P}(s_{(m,n)} | w_{(1)}, \cdots, w_{(n,T)}) = \frac{\exp(v_{s_{(m,n)}}^T v_{s_{(m,n)}})}{\sum_{s=1}^{|V_s|} \exp(v_{s_{(m,n)}}^T v_s)}
\]

(3.3)

where \(v_{s_{(m,n)}}\) is the output representation of \(s_{(m,n)}\) and \(v_{\text{word}}\) is the concatenation of the input embedding of all the words present in the sentence \(s_{(m,n)}\), with dimensionality \(T_n \times D_w\).

\[
\bar{v}_{\text{word}}^2 = [v_{w_{(n,1)}}; \cdots; v_{w_{(n,T)}}]
\]

(3.4)
<table>
<thead>
<tr>
<th>Dataset</th>
<th># Docs</th>
<th>Labels</th>
</tr>
</thead>
<tbody>
<tr>
<td>CND</td>
<td>8000</td>
<td>information retrieval, data mining, artificial intelligence, machine learning and pattern recognition, natural language and speech, computer vision, distributed and parallel computing, human-computer interaction</td>
</tr>
<tr>
<td>Wiki10+</td>
<td>19740</td>
<td>wiki, art, reference, people, culture, books, design, politics, technology, psychology, interesting, wikipedia, research, religion, music, math, development, theory, philosophy, article, language, science, programming, history, software</td>
</tr>
</tbody>
</table>

Table 3.1: Dataset Details

3.3.2 Modeling sentence-level coherence

In the next phase, we propose a novel language model which constructs the document representations from the sentence sequence present in the document. The novel task is to predict the current sentence using the embeddings of the surrounding sentences and the document embedding as features. We add the document vector in the input layer of this model, that captures the topics of the entire document in a compact form. Each document is mapped to a unique vector denoted by a column in the matrix $V_{doc}$, whose size is given by $D_d \times |V_d|$ (where $|V_d|$ is the number of unique documents).

Formally, consider $s_{(m,t-c_s)}, \ldots, s_{(m,t-1)}, s_{(m,t+1)}, \ldots, s_{(m,t+c_s)}$ as the context sentences for the target sentence $s_{(m,t)}$, appearing in the document $d_m$. The objective of our novel sentence-level language model is to maximize the following log likelihood probability.

$$
\mathcal{L}_{sent} = \sum_{d_m \in D} \log \mathbb{P}(d_m|s_{(m,1)}, \ldots, s_{(m,T_m)}) + 
\sum_{s_{(m,t)} \in d_m} \log \mathbb{P}(s_{(m,t)}|s_{(m,t-c_s)}, \ldots, s_{(m,t-1)}, s_{(m,t+1)}, \ldots, s_{(m,t+c_s)}, d_m) \tag{3.5}
$$

Here $2 \times c_s$ denotes the length of the context for the sentence sequence. The probability of observing the central sentence $s_{(m,t)}$ given the context sentences and the document is defined using the softmax function as given below.

$$
\mathbb{P}(s_{(m,t)}|s_{(m,t-c_s)}, \ldots, s_{(m,t-1)}, s_{(m,t+1)}, \ldots, s_{(m,t+c_s)}, d_m) = \frac{\exp(\bar{v}_{sent}^T v_{s_{(m,t)}})}{\sum_{s=1}^{|V_s|} \exp(\bar{v}_{sent}^T v_s')} \tag{3.6}
$$

where $v_{s_{(m,t)}}'$ is the output representation of $s_{(m,t)}$ and $\bar{v}_{sent}$ is the concatenation of the input embeddings (ignoring the central term $s_{(m,t)}$), with dimensionality $2 \times c_s \times D_s + D_d$. 

14
Similarly, we define the probability of observing the document $d_m$ given the sentences present in it as follows.

$$P(d_m | s(m, 1), \cdots, s(m, T_m)) = \frac{\exp(\bar{v}_{sent}^2 v_{d_m}')}{{\sum}_{d=1}^{|V_d|} \exp(\bar{v}_{sent}^2 v_d')}$$  \hspace{1cm} (3.8)

where $v_{d_m}'$ is the output representation of $d_m$ and $\bar{v}_{sent}^2$ is the concatenation of the input embedding of all the sentences present in the document $d_m$, with dimensionality $T_m \times D_s$.

$$\bar{v}_{sent}^1 = [v_{d_m}; v_{s(m, t-c_s)}; \cdots; v_{s(m, t-1)}; v_{s(m, t+1)}; \cdots; v_{s(m, t+c_s)}]$$  \hspace{1cm} (3.7)

3.3.3 Training details

The overall objective function of Doc2Sent2Vec is to maximize the log likelihood probability as follows.

$$\mathcal{L} = \mathcal{L}_{word} + \mathcal{L}_{sent}$$  \hspace{1cm} (3.10)

We employ stochastic gradient descent to learn the parameters, where the gradients are obtained via backpropagation [21], with fixed learning rate of 0.1. However, it takes $O(V_w)$, $O(V_s)$, $O(V_d)$ and $O(V_d)$ to compute $\nabla \log \mathbb{P}$ from Equations 3.2, 3.3, 3.6 and 3.8, which is undesirable in practice. Hence, we use hierarchical softmax [6], to facilitate faster training. The testing phase was excluded as the embeddings for all the documents in the dataset are estimated during the training phase.

3.4 Experiments and Analysis

In this section, we present the experimental results to show the effectiveness of the learned document embeddings by considering two classification tasks.

3.4.1 Dataset Description

The dataset details are displayed in Table 3.1. Citation Network Dataset (CND) [17] consists of a collection of research papers (along with abstracts) from different fields of computer science domain. Inspired by the recent work [46], we use only a sample of the original dataset to speed up the training process. We construct a dataset of 8000 papers by randomly sampling 1000 research papers from 8 different
Table 3.2: Classification Performance on CND Dataset

<table>
<thead>
<tr>
<th>Model</th>
<th>$F_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paragraph2Vec w/o WT [3]</td>
<td>0.1275</td>
</tr>
<tr>
<td>Paragraph2Vec [4]</td>
<td>0.135</td>
</tr>
<tr>
<td>Doc2Sent2Vec w/o WT</td>
<td>0.1288</td>
</tr>
<tr>
<td>Doc2Sent2Vec</td>
<td><strong>0.1513</strong></td>
</tr>
</tbody>
</table>

fields, as mentioned in Table 3.1. In the second task where we perform Wikipedia page classification, we make use of Wiki10+ dataset [14], which contains one or more social tags (along with the number of users who have annotated this tag) for each Wikipedia page retrieved from delicious.com. We find the most frequent 25 social tags and only keep those documents that contain any of these tags. It results in a collection of 19740 documents as shown in Table 3.1, with each document associated with the most voted social tag. For simplicity, we consider only the first paragraph of the Wikipedia article for learning the embeddings.

3.4.2 Experimental Setup

In all our experiments, we consider the following four models.

- **Paragraph2Vec w/o WT [3]**: Paragraph2Vec algorithm without **Word Training** (i.e. the word embedding matrix $V_w$ is freezed during training).
- **Paragraph2Vec [4]**: Extension of [3] which allows joint training of both word and document vectors.
- **Doc2Sent2Vec w/o WT**: Model discussed in Section 5.3 without word training.
- **Doc2Sent2Vec**: Model discussed in Section 5.3.

To ensure fair comparison, we empirically set $C_w$, $C_s$, $D_w$, $D_s$, $D_d$ to 5, 1, 100, 100, 100 respectively, for all the models. We lowercase all the words, remove those which occur less than 10 (15) times in the CND (Wiki10+) corpus. We use pre-trained Glove [11] word vectors trained successively on Wikipedia 2014\(^1\) and Gigaword 5\(^2\) corpus, to initialize the word embeddings ($V_w$). It is important to notice that we use a linear classifier (one-vs-rest logistic regression\(^3\)) to do prediction. It can be argued that the model performance can be improved using non-linear models, but this falls out of scope of our goal. We use 5-fold cross-validation to report the model performance for both the tasks.

\(^{1}\)http://dumps.wikimedia.org/enwiki/20140102/

\(^{2}\)https://catalog.ldc.upenn.edu/LDC2011T07

Table 3.3: Classification Performance on Wiki10+ Dataset

<table>
<thead>
<tr>
<th>Model</th>
<th>$F_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paragraph2Vec w/o WT [3]</td>
<td>0.0476</td>
</tr>
<tr>
<td>Paragraph2Vec [4]</td>
<td>0.0445</td>
</tr>
<tr>
<td>Doc2Sent2Vec w/o WT</td>
<td>0.0401</td>
</tr>
<tr>
<td>Doc2Sent2Vec</td>
<td><strong>0.0509</strong></td>
</tr>
</tbody>
</table>

3.4.3 Analysis on CND

Scientific article classification results are shown in Table 3.2. We observe that it is beneficial to learn word vectors too while training instead of merely using pre-initialised word vectors. On incorporating the learning of word vectors, the improvement of $\sim 5.88\%$ and $\sim 17.47\%$ in $F_1$ score for Paragraph2Vec and Doc2Sent2Vec respectively, justifies this claim. Moreover, we see that our model outperforms the state-of-the-art model by a significant margin of $\sim 12.07\%$. This is mainly because our model is able to exploit both word-level and sentence-level coherence to enrich the embeddings.

3.4.4 Analysis on Wiki10+

We present the Wikipedia page classification results in Table 3.3. It is interesting to see that learning the word vectors has a negative impact for Paragraph2Vec algorithm. This is shown by a decline in the $F_1$ score by $\sim 6.5\%$. We believe that the word vectors before training are semantically accurate as they are learned from the complete Wikipedia corpus. While training them again, the vectors tend to get distorted leading to poor results. It can be argued that the same trend should follow for our model when we learn the word vectors. However, the Doc2Sent2Vec results indicate that the $F_1$ improves by $\sim 26.93\%$ on learning word vectors. This illustrates that the proposed strategy of jointly exploiting sentence level and word level coherence is insensitive to the distortions generated by word vectors, resulting in robust embeddings of the Wikipedia pages. The performance improvement of $\sim 6.93\%$ over the best baselines in terms of $F_1$ score, highlights the superiority of the Doc2Sent2Vec approach.

3.5 Summary

We proposed a novel two-phase approach Doc2Sent2Vec to learn document representations in an unsupervised fashion. To this end, we introduced a novel sentence-level language model which exploits the sentence sequence present in the document. We validated the document embeddings by considering two classification tasks. Our classification results indicate the superiority of the proposed approach, thereby constituting a step towards learning accurate and rich document representations. In the next
chapter, we will see how we can use the document embeddings to introduce a novel context in author bibliographic co-authorship networks.
Chapter 4

Author2Vec - Learning Author Representations by Combining Content and Link Information

4.1 Overview

Recently, there has been an increasing interest in embedding information networks [15, 16] into low-dimensional vector spaces. The motivation is that once the embedded vector form is obtained, the network mining tasks can be solved by off-the-shelf machine learning algorithms. In an attempt to construct good representation in a scalable way, researchers have started using deep learning as a tool to analyze graphs.

DeepWalk is the state-of-the-art model, which suffers from the link sparsity problem [19] inherent in a real world information network. For example, two authors who write scientific articles related to the field ‘Machine Learning’ are not considered to be similar by DeepWalk if they are not connected. In this chapter, we aim to overcome the above mentioned problem by fusing the textual information with the link information in a synergistic fashion, for creating author representations. Our experiments on a large dataset show that harnessing the content and link information alleviates the link sparsity problem.

4.2 Problem Statement

Consider a co-authorship network $G = (V, E)$ in which each vertex represents the author and edge $e = (u, v) \in E$ represents an interaction between author $u$ and author $v$. Two authors are connected if they co-author at least one article. Let us denote the set of articles published by each author $u$ by $P_u = \{p_{u1}, \ldots, p_{uN_p}\}$, containing $N_p$ papers. For every paper, we also have the abstract and the year of publication. Then the goal of our proposed model, Author2Vec, is to learn author representations $v_u \in \mathbb{R}^d (\forall u \in V)$, where $d$ is the embedding size.
4.3 Proposed Approach

Our proposed model, Author2Vec learns the author embedding in an unsupervised way, using two types of models: Content-Info and Link-Info model, which are explained below. As the name suggests, the former model learns the textual concepts, while the latter model enriches the social dimensions further by fusing the relational concepts.

4.3.1 Content-Info Model

This model aims to capture the author representation purely by the textual content, represented by the abstracts of her papers. The model takes an author \( u \) (associated with embedding \( v_u \)) and paper \( p \) (associated with embedding \( v_p \)) as inputs and predicts whether \( u \) wrote \( p \) or not. Our training tuples consist of a set of positive input pairs (where \( p \) is a publication by the author \( u \)) and negative input pairs (where \( p \) is not a publication by the author \( u \)). The intuition to do this is to push the author representations closer to her content, and away from irrelevant content. More formally, we predict the author-paper relationship \( r_C(u, p) \), taking the value \( l \in [1, 2] \), where ‘1’ and ‘2’ denote the negative and positive input pair respectively. We predict using a neural network that considers both the angle (Eq. 4.1) and the distance (Eq. 4.2) between the input pair \( (v_u, v_p) \):

\[
h^{(x)}_C = v_u \odot v_p
\]

\[
h^{(+)}_C = |v_u - v_p|
\]

\[
h_C = \tanh(W^{(x)}_C h^{(x)}_C + W^{(+)}_C h^{(+)}_C + b^{(h)}_C)
\]

where \( W^{(x)}_C \in \mathbb{R}^{n_h \times d} \), \( W^{(+)}_C \in \mathbb{R}^{n_h \times d} \), \( b^{(h)}_C \) are the parameters of this model. Note \( n_h \) defines the hidden layer size. The usage of distance metrics, \( h^{(x)}_C \) and \( h^{(+)}_C \) is empirically motivated and similar strategies have been successfully used in Tai et al. [20]’s work to capture semantic relatedness of sentence pairs. The objective function of the Content-Info model can be written as follows.

\[
L_C = \mathbb{P}[r_C(u, p) = l] = \text{softmax}(U_C . h_C + b^{(p)}_C)
\]

where \( U_C \in \mathbb{R}^{2 \times n_h} \), \( b^{(p)}_C \) are the new parameters of this model. We learn the unknown parameters \( W^{(x)}_C \), \( W^{(+)}_C \), \( b^{(h)}_C \), \( U_C \), \( b^{(p)}_C \), \( v_u \in \mathbb{R}^d \) (i.e. author embeddings), \( v_p \in \mathbb{R}^d \) (i.e. paper embeddings) by maximizing the likelihood function in Eq. 4.4. The paper embeddings \( v_p \) are pre-initialized with output obtained from running Paragraph2Vec [3] on all the abstracts.
4.3.2 Link-Info Model

The goal of the Link-Info model is to enrich the author representations obtained from the previous model by fusing the link information. This model takes as input both the author embeddings ($v_u$ and $v_v$). Similar to the Content-Info model, the training tuples consist of positive input pairs (where $u$ has collaborated with $v$) and negative input pairs (where $u$ has never collaborated with $v$ in the training set). This setup effectively pushes the authors who share similar network structure closer in vector space from the irrelevant authors. We predict the author-author relationship $r_L(u, v)$ using a different neural network:

$$h_L^{(x)} = v_u \odot v_v \qquad (4.5)$$

$$h_L^{(+)} = |v_u - v_v| \qquad (4.6)$$

$$h_L = \tanh(W_L^{(x)} h_L^{(x)} + W_L^{(+)} h_L^{(+)} + b_L^{(h)}) \qquad (4.7)$$

where $W_L^{(x)} \in \mathbb{R}^{n_h \times d}$, $W_L^{(+)} \in \mathbb{R}^{n_h \times d}$, $b_L^{(h)}$ are the parameters of this model. The objective function of the Link-Info model can be written as follows.

$$\mathcal{L}_L = \mathbb{P}[r_L(u, v) = l] = \text{softmax}(U_L h_L + b_L^{(p)}) \qquad (4.8)$$

where $U_L \in \mathbb{R}^{2 \times n_h}$, $b_L^{(p)}$ are the new parameters of this model. We learn the unknown parameters $W_L^{(x)}$, $W_L^{(+)}$, $b_L^{(h)}$, $U_L$, $b_L^{(p)}$, $v_u \in \mathbb{R}^{d}$ (i.e. author embeddings) by maximizing the likelihood function in Eq. 4.8.

4.3.3 Training Details

We can trivially connect both the models by sharing the author embedding weights. Thus, the overall objective function of Author2Vec, maximizing the data likelihood can we written as follows.

$$\mathcal{L} = \mathcal{L}_C + \mathcal{L}_L \qquad (4.9)$$

We use stochastic gradient descent with mini-batch size of 256 and learning rate of 0.1 to learn the unknown parameters of the model.

4.4 Experiments and Analysis

We validate our learned author embeddings using two tasks: link prediction and clustering. In all the experiments, we empirically set $d$, $n_h$ to 100, 50 respectively. We use Citation Network Dataset (CND) proposed in Chakraborty [17] for both the tasks. CND contains 711810 computer science papers (along with abstracts) written by a total of 500361 authors. Each paper is tagged with one of the 24 computer science fields. We use the best experimental settings for DeepWalk reported in [16].

21
Table 4.1: Performance comparison on link prediction and clustering tasks

<table>
<thead>
<tr>
<th>Task</th>
<th>Link Prediction</th>
<th>Clustering</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model \ Metric</td>
<td>Accuracy (%)</td>
<td>NMI (%)</td>
</tr>
<tr>
<td>DeepWalk</td>
<td>81.965</td>
<td>19.956</td>
</tr>
<tr>
<td>Content-Info</td>
<td>80.707</td>
<td>19.823</td>
</tr>
<tr>
<td>Link-Info</td>
<td>72.898</td>
<td>19.163</td>
</tr>
<tr>
<td>Author2Vec</td>
<td><strong>83.894</strong></td>
<td><strong>20.122</strong></td>
</tr>
</tbody>
</table>

4.4.1 Task Description

For link prediction, we use 20 years of CND data from 1990-2009, where the last year is used for testing and the remaining as training. The positive classes are the author pairs who co-authored in the training years. For every positive pair, we choose one negative pair randomly. Test set contains the pair of authors who did not publish together in the training years, but in the test year. The resulting dataset contains 2485764 training pairs and 15342 test pairs. We use logistic regression to solve this binary classification problem. We report the accuracies for this task. CND contains manually annotated research area for each paper. For simplicity, we associate a field to each author by picking the field in which the author publishes the most. We employ K-Means algorithm (with $k=24$, denoting the number of computer science fields) using the embeddings as features and report the cross-validation scores using Normalized Mutual Information (NMI) metric.

4.4.2 Result Analysis

Results in Table 4.1 lead to the following observations:

- Using only the content information fails to perform well without linkage knowledge.

- Model which learns only using the link information gives poor results. This is because without the global content information, the author embeddings tend to be sensitive to noisy links.

- DeepWalk outperforms the previously discussed naive models, mainly due to the superiority of random walk based approach over the negative sampling approach.

- However, Author2Vec outperforms DeepWalk thanks to the fusion of the content and link information. The performance improvement of 2.35% for link prediction and 0.83% for the clustering task clearly shows the superior quality of author embeddings learned by Author2Vec.
4.5 Summary

In this chapter, we consider the problem of learning representations for authors from bibliographic co-authorship networks. Existing methods for deep learning on graphs, such as DeepWalk, suffer from link sparsity problem as they focus on modeling the link information only. We hypothesize that capturing both the content and link information in a unified way will help mitigate the sparsity problem. To this end, we present a novel model ‘Author2Vec’ \(^1\), which learns low-dimensional author representations such that authors who write similar content and share similar network structure are closer in vector space. Such embeddings are useful in a variety of applications such as link prediction, node classification, recommendation and visualization. The author embeddings we learn are empirically shown to outperform DeepWalk by 2.35% and 0.83% for link prediction and clustering task respectively. In the next chapter, we will move to Twitter, a micro-blogging social network and introduce novel context in order to learn accurate tweet embeddings.

\(^1\)Code is publicly accessible at https://github.com/ganeshjawahar/author2vec
Chapter 5

Improving Tweet Representations using Temporal and User Context

5.1 Overview

In this chapter we propose a novel representation learning model which computes semantic representations for tweets accurately. The short and noisy nature of tweets poses challenges in computing accurate latent tweet representations. We observe that Paragraph2Vec [3] which is good in computing document representation overfits when evaluated for tweets, mainly due to the short length of tweets. To overcome this problem we utilize additional context from Twitter itself. Specifically, we hypothesize that a principled usage of chronologically adjacent tweets from users’ Twitter timelines can help in significantly improving the quality of the representation. The main challenge lies in assigning appropriate attention weights to context tweets such that semantically relevant tweets receive high weights compared to less relevant ones. Consider Fig 5.1\textsuperscript{1}, where we want to learn the representation for the tweet $t(j)$. One can see that the target tweet $t(j)$ has less semantic interactions with the context tweet $t(j - 2)$. To capture this, we propose an attention based model that assigns a variable weight to each context tweet that captures the semantic correspondence between the target tweet and the context tweet. We further augment the attention model to be user-aware so that it can do well in modeling the target tweet by exploiting the rich knowledge about the user such as the way the user writes the post, and also summarizing the topics on which the user writes. Our work is closest to [7] where documents are modeled based on their word context as well as document stream context. We differ from their work in two ways: (1) they naively assume that all the documents in a stream have equal amount of semantic interactions and, (2) they ignore the knowledge of user (or document author).

5.2 Problem Statement

In this section we first introduce the notions of temporal context and attention, and then provide a formal problem statement.

\textsuperscript{1}The tweets are borrowed from Barack Obama’s Twitter timeline posted in Sep 2015.
Figure 5.1: Temporal Context with Variable Attention Example - \( t(j-1), t(j-2), t(j+1) \) and \( t(j+2) \) form the temporal context of \( t(j) \). \( \alpha \)'s denote the attention parameters of the proposed model.

**Temporal context:** Temporal context of a tweet \( t(j) \) is the set of \( C_T \) tweets posted before and after \( t(j) \) by the same user. The value \( C_T \) is a user specified parameter that defines the size of the temporal context to be considered to model a given tweet. For example, in Figure 5.1 we fix \( C_T \) as 2, the context tweets of \( t(j) \) are \( t(j-1), t(j-2), t(j+1) \) and \( t(j+2) \). Certain users may be new and so may not have sufficient number of tweets to model the context for a tweet. In such cases, our proposed model learns tweet representations using their word contexts (similar to the HDV model) only.

**Attention:** An attention value is associated with a context tweet that defines the degree of semantic similarity between the context tweet and the target tweet. The more the latent semantic interactions between the tweets, the more is the attention. We denote the attention of context tweet \( t(j-1) \) as \( \alpha(j-1) \). For instance, in Figure 5.1, the attention value of context tweet \( t(j-2) \) should be lower than that of context tweet \( t(j-1) \) with respect to target tweet \( t(j) \). In Figure 5.1, clearly \( t(j-2) \) is not talking about the topic ‘Climate Change’ and so it makes sense to have a lower attention value. In practice, we infer the degree of topical similarity automatically from the latent tweet representations being learned.

**Problem Statement:** Let the training tweets be given in the order in which they are posted. In particular, we assume that we are given a user set \( U \) of \( N_u \) tweet sequences, with each sequence \( u(k) \in U \), containing \( N_t \) tweets, \( u(k) = \{ t(1), ..., t(j), ..., t(N_t) \} \) posted by user \( u(k) \). Moreover, each tweet \( t(j) \) is a sequence of \( N_w \) words, \( t(j) = \{ w(j, 1), ..., w(j, i), ..., w(j, N_w) \} \). The problem is to learn semantic low-dimensional representations for all the tweets in the sequences in set \( U \).

### 5.3 Proposed Approach

Our model (Figure 5.2) learns tweet representations in a hierarchical fashion: learning from the words present in the tweet using word context model (Figure 5.2 (a)) along with the temporal tweets present in the user stream using tweet context model (Figure 5.2 (b)). Both the models will be discussed in detail below. Let \( w(j,i) \), \( t(j) \) and \( u(k) \) denote the embedding for a word \( i \) from tweet \( j \), tweet \( j \) and user \( u(k) \) respectively, all of which have the size ‘n’. We will discuss details about both of these models in this section.
5.3.1 Word Context Model

The goal of the word context model is to learn tweet representations which are good at predicting the words present in the tweet. The model has three layers. The first layer contains the word embeddings, $w(j, i - C_W), \ldots, w(j, i - 1), w(j, i + 1), \ldots, w(j, i + C_W)$ near the $i^{th}$ target word in tweet $j$, which denote the word context for the word $i$ along with the tweet embedding $t(j)$. Secondly, there is a hidden layer with size equal to the number of words in the vocabulary ($|V|$). The final layer is a softmax layer which gives a well-defined probability distribution over words in the vocabulary. The input to the word context model is all pairs of word context of word $i$ and tweet $t(j)$ in the corpus. The objective is to maximize the likelihood of the word $w(j, i)$ occurring given its context, i.e., $P(w(j, i) | w(j, i - C_W), \ldots, w(j, i - 1), w(j, i + 1), \ldots, w(j, i + C_W), t(j))$. Equation 5.1 represents the forward propagation step in our 1-hidden layer feed forward model, where $W_{WC}$ and $T_{WC}$ denote the additional parameters of the model.

$$\hat{y}_{|V| \times 1}(j) = \text{softmax}(W_{WC} \times \sum_{l \in \{i-C_W, i+C_W\} \setminus i} w(j, l) + T_{WC} \times t(j))$$ (5.1)

5.3.2 User + Tweet Context Model

The goal of this model is to enrich the tweet representation learned from the word context, by modeling the current tweet conditioned on its temporal context and the proposed user context. The user context makes our model user-aware by exploiting the user characteristics such as the way the user writes the post and also summarizing the topics on which the user writes. These user vectors are learned automatically from the set of tweets posted by the user through this model. As a naive solution, we can directly adopt Djuric et al. [7]’s approach and apply on the Twitter stream. As discussed in Section 5.2, this assumption is too strong for social media streams. Can we assign attention levels to the context tweets with respect to the tweet being modeled? To learn the optimal values of attention ($\alpha(j)$), we introduce the attention parameters as shown in Equation 5.2. The intuition is that semantic loss will be less if the weights of each of the temporal context tweets are learned accurately. The values of $\alpha(j)$’s can be computed as shown in Equation 5.3. The objective of this model is to maximize the likelihood of the tweet $j$ posted by user $k$ given its temporal context $(t(j - C_T), \ldots, t(j-1), t(j+1), \ldots, t(j+C_T))$ and user context $(u(k))$, which is given by $P(t(j)|t(j - C_T), \ldots, t(j-1), t(j+1), \ldots, t(j+C_T), u(k))$. Since
the tweet space can be exponentially large, we use hierarchical softmax [10] instead of normal softmax to bring down the time complexity from \(O(|T|)\) (or \(O(|V|)\) for the previous model) to \(O(\log|T|)\) (or \(O(\log|V|)\)).

\[
y_T \times 1(j) = \text{softmax}(T_T C \times \sum_{l \in \{j - C_T, \ldots, j + C_T\}} \alpha(l) \times t(l)) \quad (5.2)
\]

\[
(\alpha(j - C_T) \cdots \alpha(j - 1) \alpha(j + 1) \cdots \alpha(j + C_T)) = \text{softmax}(A[t(j - C_T); \ldots; t(j - 1); t(j + 1); \ldots; t(j + C_T);]) \quad (5.3)
\]

where the parenthesis inside the softmax function represents concatenation of all context representations ((\(2 \times C_T \times n\)) \times 1 in size). \(A\) is the additional weight matrix (of size \((2 \times C_T) \times (2 \times C_T \times n)\)) added as parameters to the model. In practice, we observe that multiple passes (‘epochs’) on the training set are required to fine tune these attention values.

### 5.3.3 Training Details

The overall objective function intertwining both the models in a hierarchical fashion to be maximized can be summarized as shown in Equation 5.4. We use the cross-entropy as the cost function between the predicted distribution \(\hat{y}(j)\) and target distributions \(t(j)\) and \(w(j, i)\), for modeling using the temporal and word context respectively. We train the model using back-propagation [21] and Adam [23] optimizer.

\[
\mathcal{L}(\theta) = \sum_{u(k) \in U} \left[ \sum_{t(j) \in u(k)} \sum_{w(j,i) \in t(j)} \log P(w(j,i)|w(j,i - C_W), \ldots, w(j, i - 1), w(j, i + 1), \ldots, w(j, i + C_W), t(j)) + \log P(t(j)|w(j, 1), \ldots, w(j, N_w)) + \log P(t(j)|t(j - C_T), \ldots, t(j - 1), t(j + 1), \ldots, t(j + C_T), u(k)) \right] + \log P(u(k)|t(1), \ldots, t(N_T)) \quad (5.4)
\]

### 5.4 Experiments and Analysis

In this section we discuss details of our dataset, experiment, and then present quantitative analysis of the proposed models.

#### 5.4.1 Task Description

We use the publicly available dataset described in Li et al. [44] for all the experiments. It contains tweets pertaining to three profile attributes (spouse, education and job) of a user. Specifically, it has a set of tweets from users’ Twitter timelines, that talk about the attribute (‘positive’ tweets) and those that do not (‘negative’ tweets). We randomly sample 1600 users from the dataset and use 70-10-20 ratio to construct train, validation and test splits. Tweet embeddings are randomly initialized while the word embeddings are initialized with the pre-trained word vectors from Pennington et al. [11].
5.4.2 Experimental Setup

We consider the binary task of predicting whether a given entity mention corresponds to particular users’ profile attribute or not. We build our model to get the tweet vector and the entity vector by computing an average of all the tweet vectors for the entity. We tune the penalty parameter of a linear Support Vector Machine (SVM) on the validation set. Note that we use a linear classifier so as to minimize the effect of variance of non-linear methods on the classification performance and subsequently help in interpreting the results. We compare our model with three baselines: (1) Paragraph2Vec [3], (2) Simple Distance model (SD): A model that assigns attention weight to the context tweet which is inversely proportional to the distance of the tweet from the target tweet, (3) HDV [7], (4) Ours (User = 0): Our model when the user context is excluded from the temporal context, (5) Ours (User = 1): Our model when the user context is included in the temporal context. We empirically set $n$ and $C_W$ to 200 and 10 respectively for all the models. In case of SD, HDV and our models, we try values in $\{1, 2, 4, 6, 8, 10, 12, 14, 16\}$ to fix the temporal context size parameter (i.e., $C_T$) which is crucial in improving the semantics of the tweet. (For example, the distance of the context tweet $t(j - 2)$ from the target tweet $t(j)$ in Figure 5.1 is 2 and hence the attention weight (i.e., $\alpha(j - 2)$) assigned to $t(j - 2)$ is 0.5 ($1/2$))

5.4.3 Comparative Analysis

From Table 5.1, we see that Paragraph2Vec overfits the validation set, resulting in poor accuracy during testing. HDV’s assumption of giving equal attention value to the temporal context also results
in lower accuracy compared with our models. SD model outperforms HDV in two tasks, which substantiates our claim against HDV’s naïve assumption for social media. Our model with user vector outperforming the baselines for Education and Job attribute classification, shows the need to consider the user characteristics while modelling his/her tweets. The poor results for Spouse task suggest that this dataset has too many topic shifts and that the user vector turned out to be less accurate. Figure 5.3 displays the F1 results for different values of $C_T$, which is a vital parameter controlling the influence of temporal context. We observe that in some cases HDV outperforms the SD model, mainly due to the inability of the SD model to utilize the context information from farther tweets which are relevant with respect to the target tweet. Our models are 19.66%, 2.27% and 2.22% better compared to the baselines for the spouse, education and job attributes respectively.

5.4.4 Impact of Variable Attention

We plot the attention mean across each position of the context tweet with respect to the epoch number. From the Figure 5.5, we see that mean attention at each context position are approximately in the ballpark. Mean attention weights vary for each context position, exhibiting no relation with respect to the increase in distance (as seen in Figure 5.4). These findings indicate the complexity of giving attention to tweets in the temporal context. Initially, we see that the mean attention weights are changing drastically indicating their sub-optimality. It is interesting to see the convergence of these weights to the optimal solution is fast (in terms of no. of epochs) in the model which uses user context when compared to the model that does not use it.
5.5 Summary

We proposed a model to learn generic tweet representations which have a wide range of applications in NLP and IR field. We discovered that the principled usage of the tweets in the temporal context is an important direction in enriching the representations. We also explored learning a novel user context vector to make our model user-aware while predicting the adjacent tweets. Through experimental analysis, we identified the cases when modeling the user characteristics help enhance the embedding quality. In the next chapter, we will discuss more about the interpretation of the semantic tweet representations so that we can understand the core properties encoded by these representations.
Chapter 6

Interpretation of Semantic Tweet Representations

6.1 Overview

Researchers in Twitter analytics are getting interesting results by applying different representation learning models for several valuable tasks such as sentiment analysis [29, 42, 46, 47], semantic textual similarity computation [30, 42], microblog retrieval [48], hashtag identification [31] and so on. However, little is known about the core properties encoded by the representations generated from these models, knowing which will allow us to make generalizable conclusions. In this chapter, we present our work which constitutes the first step in opening the black-box of vector embeddings for Twitter posts.

Essentially we ask the following question: “what are the core properties encoded in the given tweet representation?” We explicitly group the set of these properties into two categories: syntactic and social. Syntactic category includes properties such as tweet length, the order of words in it, words, slang words, hashtags, named entities, and capitalization in the tweet. On the other hand, properties such as mention count, first mention position, ‘is reply’, ‘reply time’ and repeating word from a conversation fall under the social category. We investigate the degree to which the tweet representations encode these properties. We assume that if we cannot train a classifier to predict a property based on its tweet representation, then this property is not encoded in this representation. For example, the model which preserves the tweet length should perform well in predicting the length given the representation generated from the model. Though these elementary property prediction tasks are not directly related to any downstream application, knowing that the model is good at modeling a particular property (e.g., the social properties) indicates that it could excel in correlated applications (e.g., user profiling task). In this work we perform an extensive evaluation of 9 unsupervised and 4 supervised tweet representation models, using 13 different properties.

6.2 Elementary Property Prediction Tasks

In this section we list down the set of proposed elementary property prediction tasks to test the characteristics of a tweet embedding. All the tasks are grouped into two categories: syntactic and social,
which will be explained in detail below. Note that we use a neural network to build the elementary property prediction task classifier which has the following two layers in order: the representation layer, and the softmax layer on top whose size varies according to the specific task. When there are more than one input for a task, we concatenate embeddings for each input.

6.2.1 Syntactic Tasks

In this sub-section, we discuss syntactic property prediction tasks. For each of the three tasks (a), (b) and (c) below, we randomly sample 150K, 25K and 25K tweets for training, validation and testing respectively from the Sentiment140 dataset proposed in [50].

(a) **Length Task**: This task measures the extent to which the tweet representation encodes its length. Given a tweet embedding, the task is to predict the number of words in the tweet. We use binned length, with bin size as 4 to do multi-class classification.

(b) **Content Task**: This task measures the extent to which the tweet representation encodes the identities of words present in it. Given a tweet embedding and a word embedding, the task is to predict whether the word is in the tweet or not. This is posed as binary classification task where we inject one random word not appearing in the tweet as a negative sample.

(c) **Word Order Task**: This task measures the extent to which the tweet representation preserves the word order. Given a tweet embedding, the embeddings of two words, \( w_1 \) and \( w_2 \) that appear in the tweet, the task is to predict whether the word \( w_1 \) appears before the word \( w_2 \) in the tweet or not. This is solved as a binary classification task, where the order of words are flipped to generate negative samples.

(d) **Slang Words Task**: This task measures the extent to which the tweet representation is robust to the non-standard spellings (e.g., took for took), informal abbreviations (e.g., tmrw for tomorrow), and so on, which are ubiquitous in social media. Given a tweet embedding, the embeddings of two n-grams, \( ng_1, ng_2 \), the task is to predict whether the n-gram \( ng_2 \) is the standardized form of the n-gram \( ng_1 \) (which is present in the tweet) or not. This is also posed as a binary classification task, where the n-gram \( ng_2 \) is randomly sampled to generate negative samples. We use the dataset proposed in [55] for this task.

(e) **Hashtag Task**: This task measures the extent to which the tweet representation encodes the identities of hashtags present in the tweet. Given a tweet embedding and an embedding of the word that appears in the tweet, the task is to predict whether the word is a hashtag or not. This is solved as a binary classification task, where the negative samples are generated by randomly sampling a word from the tweet which is not a hashtag. We randomly sample 150K, 25K and 25K tweets for training, validation and testing respectively from the user profiling dataset proposed in [44].

(f) **Named Entity (NE) Task**: This task measures the extent to which the tweet representation encodes the identities of the named entities present in the tweet. Given a tweet embedding and an embedding of the n-gram that appears in the tweet, the task is to predict whether the n-gram is a NE or not. This is solved as a binary classification task, where the negative samples are generated by randomly sampling
an n-gram from the tweet which is not a NE. We use the manually annotated Twitter NE corpus proposed in [39].

(g) Capitalization Count Task: This task measures the extent to which the tweet representation encodes the number of capitalized words present in the tweet. Given a tweet embedding, the task is to predict the number of words starting with a capital letter in the tweet. We use binned count, with bin size as 2 to do multi-class classification. We randomly sample 150K, 25K and 25K tweets for training, validation and testing respectively from the user profiling dataset proposed in [44].

(h) Informative Capitalization Task: Capitalization is a key orthographic feature for recognizing NE. Unlike in curated text, non-entity words in some tweets are capitalized just for emphasis and could confuse a naïve named entity recognizer. Ritter et al. [39] has proposed a strategy to annotate a tweet as having either “informative” or “uninformative” capitalization. We use the manually annotated tweet capitalization corpus proposed in their work to measure the extent to which the tweet representation encodes the capitalized word which are informative in identifying the named entity mention. In this task, we are given a tweet embedding and a embedding of the capitalized word that appears in the tweet. The task is to predict whether the word in the tweet is informative for identifying NE mention or not. This is also framed as a binary classification task. Note that all the capitalized words in a tweet belong to the same class and we do not generate negative samples for this task.

6.2.2 Social Tasks

In this sub-section, we discuss social property prediction tasks.

(i) Count Mention Task: This task measures the extent to which the tweet representation encodes the number of mentions present in it. Given a tweet embedding, the task is to predict the number of mentions (words starting with the letter ‘@’) in the tweet. We use the raw frequency and pose it as a classification problem.

(j) Position Mention Task: This task measures the extent to which the tweet representation encodes the position of the mentions in the tweet. For simplicity, we query only for the position of the first mention in the tweet. We use binned position, with bin size as 2 to do multi-class classification.

(k) Is Reply Task: This task measures the extent to which the tweet representation encodes the salient properties of a reply tweet. Given a tweet embedding, the task is to predict whether the tweet is a reply tweet or not. To generate the negative instances for this binary task, we randomly choose a tweet that is a conversation starter.

(l) Reply Time Task: This task measures the extent to which the tweet representation encodes the temporal aspects of a reply tweet. Given a tweet embedding, the task is to predict the number of minutes taken to get a reply for the tweet. For simplicity, we consider only the tweets which gets a reply within an hour. We use binned minute, with bin size as 2 to do multi-class classification.

(m) Word Repetition Task: This task measures the extent to which the tweet representation encodes the frequent words in a conversation. Given a tweet embedding and an embedding for a word, the task is to predict whether the n-gram will be used the most in the ensuing conversation thread from the tweet.

33
that is a conversation starter. We randomly choose the word that is used least in the conversation in order
to generate negative samples.

For the tasks (i) and (j) above, we randomly sample 150K, 25K and 25K tweets for training, val-
ification and testing respectively from the user profiling dataset proposed in [44]. To generate data for
the remaining tasks, we follow the crawling strategy proposed in [51] to accurately extract about ∼13M
conversation threads. We use a small subset of this corpus in this work.

6.3 Tweet Embedding Models

In this section we list down the set of models considered in the study.

6.3.1 Unsupervised

Below we list the set of unsupervised representation learning models which require an additional
classifier in general to do the final classification.

- **Bag Of Words** (BOW) [24] - This simple representation captures the tf-idf value of an n-gram.
  We pick top 50K n-grams, with the value of ‘n’ going upto 5.

- **Latent Dirichlet Allocation** (LDA) [25] - We use the topic distribution resulting by running LDA
  with number of topics as 200, as tweet representation.

- **Bag Of Means** (BOM) - We take the average of the word embeddings obtained by running the

- **Deep Structured Semantic Models** (DSSM) [37] - This is a deep encoder trained to represent
  query and document in common space, for document ranking task. We use the publicly available
  pre-trained encoder to encode the tweets.

- **Convolutional DSSM** (CDSSM) [38] - This is the convolutional variant of DSSM.

- **Paragraph2Vec** (PV) [3] - This model based on Word2Vec [18] learns embedding for a document
  which is good in predicting the words within it. We use the BOW variant with embedding size
  and window size of 200 and 10 respectively.

- **Skip-Thought Vectors** (STV) [34] - This is a GRU [41] encoder trained to predict adjacent sen-
tences in a books corpus. We use the recommended combine-skip (4800-dimensional) vectors
  from the publicly available encoder.

- **Tweet2Vec** (T2V) [31] - This is a character composition model working directly on the character
  sequences to predict the user-annotated hashtags in a tweet. We use publicly available encoder,
  which was trained on 2M tweets.
• **Siamese CBOW (SCBOW)** [30] - This model uses averaging of word vectors to represent a sentence, and the objective and data used here is same as that for STV. Note that this is different from BOW because the word vectors here are optimized for sentence representation.

### 6.3.2 Supervised

Below we list the set of supervised representation learning models which are capable of performing end-to-end classification.

- **Convolutional Neural Network (CNN)** - This is a simple CNN proposed in [35].

- **Long Short Term Memory Network (LSTM)** [40] - This is a vanilla LSTM based recurrent model, applied from start to the end of a tweet, and the last hidden vector is used as tweet representation. We use the optimal hyper-parameter settings proposed in [20].

- **Bi-directional LSTM (BLSTM)** [40] - This extends LSTM by using two LSTM networks, processing a tweet left-to-right and right-to-left respectively. Tweet is represented by concatenating the last hidden vector of both LSTMs. We use the optimal hyper-parameter settings proposed in [20].

- **FastText (FT)** [36] - This is a simple architecture which averages the n-gram vectors to represent a tweet, followed by the softmax in the final layer. This simple model has been shown to be quite effective for text classification task.

### 6.4 Experiments

In this section we perform an extensive evaluation of all the models in an attempt to find the significance of different representation models. Essentially we study every model (with optimal settings reported in the corresponding paper) with respect to the following five perspectives.

1. **Property prediction task accuracy** - This test identifies the model with the best F1-score for each elementary property prediction task.

   (a) *Best of all in:* Property prediction tasks for which this model has outperformed all the other models.

   (b) *Best of unsupervised approaches in:* Property prediction tasks for which this model has outperformed all the other unsupervised models.

   (c) *Best of supervised approaches in:* Property prediction tasks for which this model has outperformed all the other supervised models.

2. **Property prediction task accuracy versus Tweet length** - This test helps to compare the performance of the model for shorter and longer tweets.
Table 6.1: Elementary Property Prediction Task F1-Score (%) - Performance Comparison

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Unsupervised</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BOW</td>
<td>37.83</td>
<td>97.37</td>
<td>60.36</td>
<td>78.13</td>
<td>99.28</td>
<td>89.66</td>
<td>37.59</td>
<td>72.39</td>
<td>87.02</td>
<td>75.2</td>
<td>78.14</td>
<td>35.98</td>
<td>86.92</td>
</tr>
<tr>
<td>LDA</td>
<td>25.11</td>
<td>97.72</td>
<td>60.62</td>
<td>76.82</td>
<td>99.35</td>
<td>97.24</td>
<td>55.25</td>
<td>68.66</td>
<td>69.31</td>
<td>36.85</td>
<td>60.12</td>
<td>28.03</td>
<td>91.71</td>
</tr>
<tr>
<td>BOM</td>
<td>47.64</td>
<td>98.67</td>
<td>61.25</td>
<td>75.26</td>
<td>99.33</td>
<td>98.06</td>
<td>59.1</td>
<td>73.13</td>
<td>74.26</td>
<td>45.64</td>
<td>66.25</td>
<td>28.43</td>
<td>92.26</td>
</tr>
<tr>
<td>DSSM</td>
<td>57.76</td>
<td>98.57</td>
<td>59.01</td>
<td>76.89</td>
<td>99.33</td>
<td>97.16</td>
<td>69.57</td>
<td>71.64</td>
<td>78.86</td>
<td>52.46</td>
<td>76.47</td>
<td>29.08</td>
<td>91.93</td>
</tr>
<tr>
<td>CDSSM</td>
<td>47.75</td>
<td>98.09</td>
<td>57.66</td>
<td>69.8</td>
<td>99.35</td>
<td>97.41</td>
<td>62.42</td>
<td>68.66</td>
<td>75.48</td>
<td>44.38</td>
<td>73.92</td>
<td>28.49</td>
<td>92.47</td>
</tr>
<tr>
<td>PV</td>
<td>13.58</td>
<td>94.9</td>
<td>60.92</td>
<td>76.09</td>
<td>98.02</td>
<td>65.11</td>
<td>33.14</td>
<td>70.89</td>
<td>45.98</td>
<td>21.3</td>
<td>54.68</td>
<td>27.58</td>
<td>90.71</td>
</tr>
<tr>
<td>STV</td>
<td>71.85</td>
<td>98.85</td>
<td>57.7</td>
<td>76.66</td>
<td>99.32</td>
<td>97.92</td>
<td>72.28</td>
<td>70.89</td>
<td>98.94</td>
<td>68.46</td>
<td>96.41</td>
<td>29.25</td>
<td>92.82</td>
</tr>
<tr>
<td>T2V</td>
<td>73.58</td>
<td>98.36</td>
<td>60.62</td>
<td>62.34</td>
<td>99.32</td>
<td>92.93</td>
<td>71.81</td>
<td>82.84</td>
<td>86.5</td>
<td>66.29</td>
<td>95.73</td>
<td>31.59</td>
<td>89.76</td>
</tr>
<tr>
<td>SCBOW</td>
<td>32.13</td>
<td>97.94</td>
<td>58.39</td>
<td>74.24</td>
<td>99.35</td>
<td>97.79</td>
<td>43.32</td>
<td>69.4</td>
<td>70.62</td>
<td>33.59</td>
<td>60.38</td>
<td>28.39</td>
<td>92.49</td>
</tr>
<tr>
<td>Super.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CNN</td>
<td>59.48</td>
<td>97.71</td>
<td>61.13</td>
<td>77.42</td>
<td>99.31</td>
<td>91.38</td>
<td>86.32</td>
<td>88.81</td>
<td>91.04</td>
<td>77.15</td>
<td>92.66</td>
<td>31.73</td>
<td>87.28</td>
</tr>
<tr>
<td>LSTM</td>
<td>99.79</td>
<td>97.39</td>
<td>60.74</td>
<td>76.24</td>
<td>99.28</td>
<td>90.36</td>
<td>89.23</td>
<td>61.19</td>
<td>89.49</td>
<td>80.26</td>
<td>92.39</td>
<td>28.46</td>
<td>85.58</td>
</tr>
<tr>
<td>BLSTM</td>
<td>98.72</td>
<td>97.47</td>
<td>60.85</td>
<td>80.52</td>
<td>99.28</td>
<td>90.89</td>
<td>89.73</td>
<td>75.37</td>
<td>89.83</td>
<td>91.2</td>
<td>92.76</td>
<td>27.99</td>
<td>86.43</td>
</tr>
<tr>
<td>FT</td>
<td>24.56</td>
<td>92.15</td>
<td>60.06</td>
<td>67.48</td>
<td>89.11</td>
<td>78.89</td>
<td>76.14</td>
<td>57.46</td>
<td>83.19</td>
<td>61.31</td>
<td>74.08</td>
<td>28.35</td>
<td>70.58</td>
</tr>
</tbody>
</table>

(a) **Positively correlated tasks**: Property prediction tasks for which the performance of the model increases as tweet length increases.

(b) **Negatively correlated tasks**: Property prediction tasks for which the performance of the model decreases as tweet length increases.

3. **Embedding Norm (average) versus Tweet length** - This task agnostic test is only for unsupervised algorithms.

(a) **Positively correlated**: Does the average embedding norm increase as tweet length increases?

(b) **Negatively correlated**: Does the average embedding norm decrease as tweet length increases?

4. **Representation size versus Property prediction task accuracy** - This test captures the sensitivity of each supervised model with respect to the embedding size. We consider only the supervised model for this setup as we use the pre-trained version for most of the unsupervised models.

(a) **Invariant tasks**: Property prediction tasks for which the model performance is invariant with increase in representation size.

(b) **Positively correlated tasks**: Property prediction tasks for which the model performance increases with increase in representation size.

(c) **Negatively correlated tasks**: Property prediction tasks for which the model performance decreases with increase in representation size.
5. **Sensitivity of Property prediction task to Word Order** - This refers to the setting where the words in the tweets are randomly ordered. This helps in testing the extent to which a model relies on the word ordering properties of the natural language.

(a) *Invariant tasks*: Property prediction tasks for which the model performance does not decline even when the words in the tweets are randomly reordered.

(b) *Significantly deviant tasks*: Property prediction tasks for which the model performance declines significantly when the words in the tweets are randomly reordered.

### 6.5 Results and Analysis

Fine-grained analysis of various supervised and unsupervised models discussed in Section 6.3, across various dimensions discussed in Section 6.4, is discussed in detail in this section. The codes used to
conduct our experiments are publicly accessible at: https://github.com/ganeshjawahar/fine-tweet.

### 6.5.1 Property Prediction Task Accuracy

We summarize the results of property prediction tasks in Table 6.1. Length prediction turns out to be a difficult task for most of the models. Models which rely on the recurrent architectures such as LSTM, STV, T2V have sufficient capacity to perform well in modeling the tweet length. Also BLSTM is the best in modeling slang words. BLSTM outperforms the LSTM variant in all the tasks except ‘Content’, which signifies the power of using the information flowing from both the directions of the tweet. T2V which is expected to perform well in this task because of its ability to work at a more fine level (i.e., characters) performs the worst. In fact T2V does not outperform other models in any task, which could be mainly due to the fact that the hashtags which are used for supervision in learning tweet representations reduces the generalization capability of the tweets beyond hashtag prediction. Prediction tasks such as ‘Content’ and ‘Hashtag’ seem to be less difficult as all the models perform nearly optimal for them. The superior performance of all the models for the ‘Content’ task in particular is unlike the relatively lower performance reported for in [33], mainly because of the short length of the tweets. The most surprising result is when the BOM model turned out to be the best in ‘Word Order’ task, as the model by nature loses the word order. This might be due to the correlation between word order patterns and the occurrences of specific words. BOM has also proven to perform well for identifying the named entities in the tweet. For the capitalization tasks such as ‘Capt. Count’ and ‘Info. Capt’, we observe the supervised models to perform better than the unsupervised models.

STV is good for most of the social tasks including ‘Count Mention’, ‘Is Reply’ and ‘Word Repeat’. We believe the main reason for STV’s performance is two-fold: (a) the inter-sentential features extracted from STV’s encoder by the prediction of the surrounding sentences in the books corpus contains rich social elements that are vital for social tasks (e.g., user profiling), (b) the recurrent structure in both the encoder and decoder persists useful information in the memory nicely. The second claim is further substantiated by observing the poor performance of SCBOW whose objective is also similar to STV, but with a simpler architecture (i.e., word vector averaging). In future it would be interesting to create such a model for Twitter conversations or chronologically ordered topical tweets so as to directly capture the latent social features from Twitter.

### 6.5.2 Sensitivity to Tweet Length

This setup captures the behavior of the model with the increase in the context size, which is defined in terms of number of words. For tasks such as ‘Word Order’, ‘Pos of the first Mention’ and ‘Capt. Count’, we see the performance of all the models (Figure 6.1) to be negatively correlated with the tweet length. On the other hand, there is no correlation between the tweet length and the performance of all the models for the tasks such as ‘Slang Words’, ‘Content’, ‘Hashtag’, ‘NE’, ‘Info. Capt’ and ‘Is Reply’. For
social tasks such as ‘Is Reply’, ‘Reply Time’ and ‘Word Repeat’, we see a positive correlation between the tweet length and the performance of all the models (Figure 6.1). This finding is intuitive in social media analysis where additional context is mostly helpful in modeling the social behavior.

### 6.5.3 Embedding Norm vs Tweet Length

Figure 6.2 plots the average embedding norm of all the models with respect to the number of words in the tweet. It is evident that the norm of the bag of words models such as BOM and SCBOW are negatively correlated with the tweet length. We argue that this helps the bag of words models persist the length information and perform competitively with other sophisticated models. The norm of most of the unsupervised models such as BOW, LDA, DSSM, CDSSM and PV are positively correlated with the tweet length. We find the norm of T2V embeddings to be very high, which could be mainly due to the fact that its final embedding is compositioned from several character-level embeddings (in worst case, it could be 140 character length tweet). On the other hand, we find the STV model to be the only model to be robust to this setup, i.e., the average embedding norm remains constant regardless of the size of the tweet. This may be the reason why STV is regarded as a general model applicable to text data from different domains.

### 6.5.4 Representation Size vs Property Prediction Task Accuracy

In representation learning, low embedding size results in a poor model performance as the model does not have enough capacity (‘underfits’) to retain information. On the other hand, high embedding size also results in poor performance as the model has redundant bits of information (‘overfits’) which has a negative effect. The optimal strategy mostly is to do grid search for the size that gives superior performance. Specifically we build models with the embedding size from \{10, 25, 50, 100, 200\}. Figure 6.3 displays the plots for all the models for this setup. We find all the supervised models except FT to be positively correlated with the representation size for most of the property prediction tasks. We discover that FT which relies on a simple operation of word vector average to represent a tweet is invariant to the representation size. This result is surprising as FT could potentially yield good performance.
Figure 6.3: Model Performance for varying representation size and word order - Only for supervised models. The suffix ‘P’ indicates that the model is built and tested on tweets whose words are randomly ordered.
6.5.5 Sensitivity of Property prediction task to Word Order

This test essentially captures the importance of “natural word order”. LDA has proven to be invariant to the reordering of the words in the tweet for most of the tasks. This result is not surprising as LDA considers each word in the tweet independently. Figure 6.3 displays the plots for all the supervised models in this setup. CNN, LSTM and BLSTM rely on the word order significantly to perform well for most of the prediction tasks.

6.5.6 A comment on BOW vs Paragraph2Vec

We observed that Paragraph2Vec which has shown to be effective in generating general purpose document embeddings performed poorly for all of the property prediction tasks. It has to be noted that this model has not been extensively evaluated for tweets in literature. We wish to fill this void by comparing Paragraph2Vec with BOW for a wide variety of benchmarked tweet corpora. Specifically, we evaluate the models for five tasks: (1) predict whether the sentiment of tweet is positive, negative or neutral (SA) [53], (2) predict the entity the tweet belongs to [45] (EI), (3) predict the priority of the topic the tweet belongs to [45] (TP), (4) predict the day of the weather referred in the tweet [54] (W), and (5) predict the kind of the weather referred in the tweet [54] (K). Table 6.2 reports the scores of the best performing Paragraph2Vec with the variant (BOW or Distributed Memory) and representation size ({10, 25, 50, 100, 200}) tuned using the validation set. From the results, we find that Paragraph2Vec overfits the dataset due to the short length of the tweet, resulting in poor performance for all the tasks compared to BOW. This finding also suggests the social media researchers to consider BOW model as an important baseline while comparing with the state-of-the-art tweet representation models.

6.6 Summary

This work proposed a set of elementary property prediction tasks to understand different tweet representations in an application independent, fine-grained fashion. The open nature of social media not only

<table>
<thead>
<tr>
<th>Model/Task</th>
<th>SA</th>
<th>EI</th>
<th>TP</th>
<th>W</th>
<th>K</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOW</td>
<td>62.77</td>
<td>90.44</td>
<td>64.62</td>
<td>80.80</td>
<td>82.06</td>
</tr>
<tr>
<td>Paragraph2Vec</td>
<td>52.59</td>
<td>36.05</td>
<td>55.73</td>
<td>77.53</td>
<td>30.34</td>
</tr>
</tbody>
</table>
poses a plethora of opportunities to understand the basic characteristics of the posts, but also helped us draw novel insights about different representation models.
Chapter 7

Conclusions and Future Work

7.1 Conclusion

In this thesis we proposed solutions to solve two main challenges: context insufficiency and interpretability, present with the current representation learning models employed in the NLP / IR area. Specifically, we discovered novel solutions to overcome the sparsity issues with the document, author and tweet representation models by augmenting them with additional novel contexts. This helped in improving the performance of the models in several downstream applications. We also proposed a framework to interpret the tweet representations in a fine-grained fashion. This helped us understand the core properties encoded by these representations.

7.2 Future Work

Some of the future directions and research problems which emerge out of this thesis are as follows:

- **Exploiting other sub-divisions of a document**: Doc2Sent2Vec computes accurate document representation using word-level and sentence-level contexts. In future, we plan to extend the current approach to a general multi-phase approach where every phase corresponds to a logical sub-division of a document like words, sentences, paragraphs, subsections, sections and documents.

- **Exploiting context from a document stream**: It will be interesting to investigate how the document embeddings that are learned through the Doc2Sent2Vec approach can be enhanced by considering the document sequence in a stream such as news click-through streams [7].

- **Exploiting other contexts in the network**: Author2Vec computes accurate author representations using content and link contexts. In future, we plan to extend the model to capture other useful contexts such as journal / conference context (set of papers published by the journal / conference), paper-meta context (meta-info of papers like keywords), citation context (set of papers cited by the paper) and author-meta context (meta-info of authors like age and country).
• **Extending to weighted graphs**: Author2Vec works with unweighted graph as it assumes the weight of every edge in the graph to be 1. In future, we plan to extend the model for weighted graphs, where edge weights indicate the number of papers co-authored.

• **Extension to capture global network information**: Author2Vec’s link-info model works with the local network information. In future, it will be interesting to explore the impact of capturing the global network information in enhancing the quality of the embeddings.

• **Exploiting conversational and topical context**: Our tweet representation model used the content and temporal information to learn the tweet embedding. To the best of our knowledge, there is no related work which exploits the signals from the conversational context (set of surrounding tweets in a conversation) or from the topical context (set of chronologically ordered topically relevant tweets).

• **Exploiting the Twitter users’ social features**: Twitter provides a platform for the users to interact with other users. To the best of our knowledge, there is no related work which exploits the profile attributes like profile picture, user biography and set of followers, and social interactions like retweet context (set of surrounding tweets in a users’ retweet stream) and favorite context (set of surrounding tweets in a users’ favorite tweet stream).

• **Interpreting the node embeddings**: In this thesis we proposed a framework for interpreting the tweet embeddings in a fine-grained fashion. With the surge in the interest in applying representation learning solutions to analyze graphs, it would be really interesting to interpret the node embeddings. To the best of our knowledge, there is no work in this research direction to understand the network properties encoded in node embeddings.

To summarize, this thesis proposed solutions to handle the context insufficiency and interpretability problem with the representation learning models. However, we are aware that these solutions are barely scratching the surface of the growing field of representation learning and it would surely serve as a building block to help the NLP and IR practitioners to further the technology in their respective fields.
Related Publications

- Ganesh J, Manish Gupta, Vasudeva Varma, **Doc2Sent2Vec: A Novel Two-Phase Approach for Learning Document Representation**, In 2016 ACM International Conference on Special Interest Group on Information Retrieval (SIGIR), Pisa, Italy.

- Ganesh J, Soumyajit Ganguly, Manish Gupta, Vikram Pudi, Vasudeva Varma, **Author2Vec: Learning Author Representations by Combining Content and Link Information**, In 2016 ACM International Conference on World Wide Web Conference (WWW), Montreal, Canada.


Bibliography


[54] https://www.kaggle.com/c/crowdflower-weather-twitter

[55] https://noisy-text.github.io/norm-shared-task.html