3-26-2020 10:30 AM

# Investigating Citation Linkage as a Sentence Similarity Measurement Task using Deep Learning

Sudipta Singha Roy
*The University of Western Ontario*

Supervisor
Mercer, Robert E.
*The University of Western Ontario*

Graduate Program in Computer Science
A thesis submitted in partial fulfillment of the requirements for the degree in Master of Science

# Abstract

Research publications reflect advancements in the corresponding research domain. In these research publications, scientists often use citations to bolster the presented research findings and portray the improvements that come with these findings, at the same time, to make the contents more understandable to the audience by navigating the flow of information. In the science domain, a citation refers to the document from where this information originates but doesn't specify the text span that is actually being cited. A more precise reference would indicate the text being referenced. This thesis develops a framework which can create a linkage between the citing sentences from the ongoing research article and the related cited sentences from the corresponding referenced documents. This citation linkage problem has been modeled as a semantic relatedness task where given a citing sentence the framework pairs this citing sentence with each sentence from the reference document and then tries to determine which sentence pair is semantically similar and which pair is not. Construction of the citation linkage framework involves corpus creation and utilizing deep-learning models for semantic similarity measurement.

**Keywords:** citation linkage, textual semantic relatedness, text classification, dataset creation

# Lay Abstract

Research papers refer to some other research documents to bolster the proposed ideas in the ongoing paper or when ideas from those documents are used. Some times, these references are made to help the readers to build a good background over the topics. This referencing is done by means of citation. After discussing any idea a citation is made. The span of the citation may contain only one or more than one sentence. However, for the readers, this citation sentences and the citations are nothing but a link to the referenced paper. It doesn't give any specific link or hint about which section of paragraph in the reference paper is actually being referred. So, the readers have to read the whole referenced document. Citation linkage intents to reduce this burden from the readers. The idea of citation linkage is to build a framework which given a citation sentence and the cited research document tries to detects those sentences which are being referred by the citation sentence in the ongoing paper by means of checking the semantic similarity of every sentence of the referred research document against the citation sentence.

# Acknowledgements

At first, I would I like to express my sincere gratitude to my honorable thesis supervisor Dr. Robert E. Mercer for his guidance, ideas, expertise, support, feedback and encouragement from the very beginning to the end of my journey. His feedback and encouragement have motivated me when I faced a lot of challenges. His guidance and expertise have helped me come out of various difficulties and challenges I have faced while conducting the experiments. Special thanks to Mahtab Ahmed and Felipe Urra for their contributions and helps in this work.

I am grateful to the Department of Computer Science of The University of Western Ontario for providing me with the Graduate Research Scholarship.

In the end I would like to thank my parents and wife for their support and having faith in me. Without their mental support the whole process would have been very complicated for me.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Different types of written documents have different formats, writing patterns and serve different objectives. A research article can reflect research trends, a new invention, or perspectives to solve a problem in a particular domain. While writing a research document, the author discusses previous research that is either prominent to solve the same problem or has influenced the author's ideas presented in the ongoing research paper. This referencing to some other document while writing a research article is called a *citation* [26]. Thus, citations create links between various research articles. Usage of citations reduces the writing overload of authors as they don't have to write down the same thing. At the same time, it helps readers achieve some background knowledge over a topic which may be unknown to them but required to understand the ideas in the ongoing article. Citations assist researchers in other ways. For example, in 1964 the idea of citation indexing was introduced [18]. Citation indexes contain the references found in research documents in many scientific domains. Citation investigation based bibliometrics are utilized to evaluate significance of any research work [19]. In 2000, Garzone and Mercer [20] proposed a way to identify the internal purpose of a particular citation. As well, modern applications such as multiple document summarization [51] and argumentation mining across multiple research articles [41] use these citation links.

In the case of scientific research articles, a citation refers to the document from where the idea stated in the citing sentence originates. However, a citing sentence typically refers to a small portion of the referenced document. This referred to text span can be the summary of the referred to paper's methodology, or the results and findings, or the analysis of such. This specific portion of text can be either single or multiple paragraphs or sentences. Citations help the authors to be very specific about the ideas they want to share from some other research documents, but at the same time, it reduces the writing overhead for the authors and helps them maintain the constraints of page limits imposed by the publication authorities [26]. If it were possible to pull out that specific text span from the reference document, it would be

advantageous for applications such as those mentioned above. Additionally, it would reduce the burden of the readers having to read the complete cited document to find the piece that is being cited. These objectives are the motivation for the topic of this thesis: to establish a relationship between the citation sentence and its corresponding reference sentences from the cited paper. This task is called *citation linkage* [26]. In this study, citation linkage is modelled as a textual semantic relatedness measurement task and the text span is delimited to a single sentence. Hence, this is why the citation linkage problem is formulated as a textual matching operation between a citation sentence and every sentence in the corresponding cited paper. This thesis investigates the ways to determine citation linkage using different unsupervised and supervised deep-learning methods.

A citation establishes a semantic link between the citing and the cited paper and thus helps to develop a body of knowledge from the previous research findings that bolsters the ideas presented in the ongoing paper. For this the author can either explicitly use the words and formulas from the cited paper or can refer to inherent domain information. In both ways, both the citing and the cited paper share some common domain ideas and information which is introduced in the cited paper. Table 1.1 shows a few examples of citation sentences and their corresponding reference sentences from a cited paper. Example 1 gives the sentence pair where the citation sentence is a paraphrase of the cited sentence. Both of the sentences contain common words in a different order. In the second example, the citing sentence replaces the term "pH4" by "extremely acidic environment". To establish a linkage between these two terms a mapping would be required between the pH scales and the acidic condition. From the sentence pair in Example 3, it can be observed that the citing sentence interprets the information from the target sentence. From these examples it is clear that for proper linkage between citation and cited sentence, the proper mapping is essential from the word to the sentence level. As the final models used for the textual semantic relatedness measurement are deep-learning models, proper word and sentence embedding techniques are required prior feeding the data to these models. Furthermore, while measuring semantic relatedness between two sentences, only a few words rather than all the words in the sentences play the vital role. That's why the attention mechanism is used to give proper importance over individual words in the sentences.

The objective of this thesis is to establish a relationship between the citation sentence and its corresponding reference sentences from the cited paper, a task called *citation linkage*. In this study, citation linkage is modelled as a textual semantic relatedness measurement task and the text span is delimited to a single sentence. Hence, this is why the citation linkage problem is formulated as a textual matching operation between a citation sentence and every sentence in the corresponding cited paper.

The major contributions of this thesis work are building a corpus for citation linkage task

Table 1.1: Sample citation and corresponding target reference sentence (source: Houngbo [26]).

| | | |
|---|---|---|
| **Example 1** | Citation Sentence | Formalin fixation, the most widely used fixative in histopathology, has many advantages such as the ease of tissue handling, the possibility of long-term storage, an optimal histological quality and its availability in large quantities at low price. [28] |
| | Target Sentence | The advantages of formalin fixation are the ease of tissue handling, the possibility of long-term storage of wet material, and its low price. [30] |
| **Example 2** | Citation Sentence | Sample DNA is often damaged by exposure to formaldehyde and a potentially extremely acidic environment. [60] |
| | Target Sentence | However, DNA is relatively stable in mildly acidic solutions, but at around pH 4 the β glycosidic bond in the purine bases are hydrolysed. [5] |
| **Example 3** | Citation Sentence | Different PCR buffer systems and/or different Taq polymerases may yield different real time PCR results. [28] |
| | Target Sentence | A significant difference can be seen between the results from the different DNA polymerase-buffer systems. [62] |

containing more than sixty thousand sentence pairs from the biomedical domain, developing a method for cleaning and preprocessing sentences from different biomedical domains and building a framework to determine the appropriate cited sentences from a cited paper given a citation sentence. The new corpus, data cleaning tool, and citation linkage tool are available on GitHub (https://github.com/sudipta90/CitationLinkage).

The rest of the thesis is structured as follows: Chapter 2 reviews some related literature on the four main concepts used in the research: the citation linkage task, word embedding, sentence embedding, and the attention mechanism used in neural architectures. Chapter 3 describes the data preprocessing, dataset creation techniques along with the overall architectures used for the citation linkage task. The experimental results are elaborately reported and analyzed in Chapter 4. In the end, the summary of the citation linkage work that has been accomplished in this thesis is presented along with the shortcomings and direction for future work in this domain.

# Chapter 2

# Literature Review

This chapter discusses research that has been conducted that directly concerns the citation linkage task or research that will be useful for our approach to addressing this task.

Much research can be found for the citation analysis task in experimental sciences [39, 38, 6, 27, 44, 53, 19, 20]. Given a citing sentence, citation analysis tries to determine the section (Introduction, Methods, Results, and Discussion) of the cited paper that is being referred to by the citing sentence. However, most of this research cannot indicate the specific sentences or the paragraphs from the cited paper that are being cited. On the other hand, few works are available for the citation linkage task. The methods provided are discussed in Section 2.1.

Because this thesis formulates the citation linkage problem as a textual semantic similarity measurement task modelled using deep learning, various deep learning models are discussed in Section 2.2. As the textual semantic similarity measurement depends on words and sentences, word and sentence embedding techniques form a major portion of this discussion. These techniques are discussed in Sections 2.2.2 and 2.2.3. Furthermore, not all the words in a sentence contribute equally to build the semantic meaning of the sentence. Different attention mechanisms have been introduced in recent times to put more focus on the important words in a sentence. Utilizing attention mechanisms in textual semantic similarity measurement tasks has improved the performance to some extent. Section 2.2.4 will discuss the attention mechanisms that will be used for the citation linkage task.

## 2.1 Citation Linkage Research

In 2017, Houngbo and Mercer [25] developed a framework to detect cited sentences given a citation sentence. For this task , they built a small corpus which was annotated by a domain expert. The annotation was done over sentence pairs containing 23 citation sentences and 3857 candidate cited sentences. All the research papers chosen for this task were from the

biomedical domain. Each sentence pair in this dataset is labeled with an integer number in the range from 0 to 5 where 0 indicates the sentences are not similar at all and 5 indicates highest similarity between the citation and cited sentences. For this task, they used different machine learning models. However, the accuracy they achieved was low. They reported capturing only 48.5% positive samples correctly. Here the positive samples are those which are annotated with similarity indexes 4 and 5.

In 2018, Li et al. [33] applied a ruled-based approach to determine the citation linkage between citation and cited sentence pairs. For this task, they used textual semantic similarity at the sentence level. To compute the semantic similarity between citation and candidate citing sentences, they computed inverse document frequency (idf)and Jaccard similarity. Later, they trained Word2Vec [43] to get 200 dimensional word vectors and represented the sentence as the concatenation of the word vectors in the sentence. To calculate the similarity between the sentences, they then computed the cosine similarity between the sentence vectors. Later in their following work, they ran the convolutional neural network (CNN) over the sentence representations to generate better feature representations and then computed cosine similarity between the citation and cited sentences [34]. They conducted these experiments over computational linguistics literature. The performance of their model in the case of analyzing citation linkage for biomedical research articles is presented in Table 4.3 in Chapter 4.

## 2.2 Neural Net Research

In recent times, the introduction of deep learning models in the natural language processing domain has boosted the performances for almost all possible applications. Simple deep neural nets have one limitation. They are not capable of handling sequential data whereas human conversations and sentences are sequential data. While reading one document, a person understands the meaning of the current content based on the previous words instead of forgetting everything of the previous section of the document and thinking from scratch at each time step. Recurrent neural networks (RNN) have the ability to preserve the information from the past while considering current content. Recently, RNN based models have been used for different NLP tasks like semantic similarity analysis, sentiment analysis, language model generation, etc. Still, deep learning models can't work with string data directly. They require the vector form for both words and sentences. To solve this issue, different word and sentence embedding techniques have been developed in the last decade. Furthermore, in case of NLP, the semantics of a sentence or a document doesn't depend equally on all of the words in the sentence. An attention mechanism is applied in these cases to focus on different portions of the texts. This section discusses different recurrent neural network models, word and sentence embedding

techniques, and attention mechanisms using deep learning architectures.

### 2.2.1 Recurrent Neural Networks

In the case of natural languages, the meaning of a sequential text doesn't depend on the last word only, rather words at different positions help to develop the inherent sentiment or semantics of the text span. Traditional deep neural nets cannot work with this persistence of information. However, recurrent neural networks (RNNs), with the inner loop inside the structure, have the ability to process continuous data. Figure 2.1 portrays the working principle of the traditional RNN module.

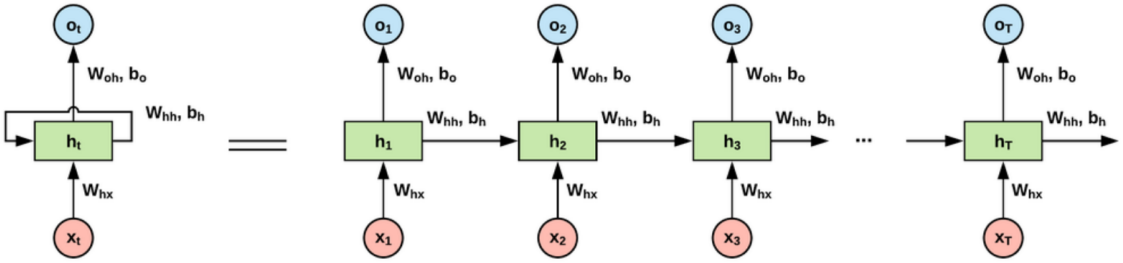



Figure 2.1: Unrolled recurrent neural network (source: easy-tensorflow [17]).

A simple RNN module looks at an input $x_t$ at a time step $t$ and outputs a hidden state representation $h_t$ and forwards this information to the next time step. Thus, the RNN propagates information from the past. An RNN can be viewed as multiple identical neural networks placed in sequence to process the information at different time steps, where each module passes the information to its following one. Hidden layers in an RNN share the same bias and weight matrices. The working principle of a simple RNN module is explained by Eqn. 2.1 and 2.2 [47].

$$h_t = f(W_{hh}h_{t-1} + W_{hx}x_t) \tag{2.1}$$

$$o_t = W_{oh}h_t \tag{2.2}$$

where, $o_t$ is the output state at time step $t$, $h_t$ is the current hidden state at time step $t$, $h_{t-1}$ is the hidden state from the previous time step forwarded to the current time step and $f$ is the activation function. All $W$s represent different weights like $W_{hh}$ is the weight matrix between hidden states of two time steps, $W_{hx}$ is the weight between input and hidden states, and $W_{oh}$

is the weight matrix between hidden and output states. Initially all the weights and biases are initialized randomly and gradually they are updated via backpropagation. For an RNN, this backpropagation goes back to each time step. This technique is called backpropagation through time (BPTT).

This chain-like architecture allows the RNNs to work with sequential data. As a result, RNNs are being used successfully in speech recognition, machine translation, language modeling, image captioning, etc. in recent times. However, the simple RNN model suffers from two issues while working with long sequential data: the vanishing gradient and the exploding gradient. In the vanishing gradient problem, the gradient becomes very small preventing the weights from changing the values and may stop the training of the neural network. For example, the gradients of run-of-the-mill activation functions like the sigmoid function remain in the range (0, 1). During the backpropagation gradients are computed by the chain rule. While computing the gradients for the $n$th time step, these $n$ small gradient values have been multiplied with each other thus generating a very small number. With very large values of $n$, the gradient decreases exponentially and in the end, makes the convergence of the model very slow [23]. For exploding gradients, error gradients are accumulated during backpropagation and in the end can result in very large gradients. This situation makes the network unstable as the network weights are changed at a too high rate than it should be, preventing convergence of the network. If the gradients are too big, the network weights become very large and instead of generating real values the network may output "not a number" values [11]. To solve these issues, a few different models like long short-term memory (LSTM), gated recurrent units (GRU), etc. have been proposed.

### Long Short-Term Memory

Long Short-Term Memory (LSTM) [24] is an RNN model that has been upgraded to work with long term dependencies. Like the simple RNN, an LSTM maintains the chain-like structure. But, instead of working with a simple neural network layer only like RNN, it consists of four layers to preserve the important information in its memory from the past. The architecture of an LSTM is illustrated in Figure 2.2.

The key component of an LSTM cell is its cell state ($c_t$) which acts as the memory of the network. The first step of an LSTM cell is to decide which information to preserve and which to remove. A sigmoid layer named the "forget gate" is responsible to make this decision. It takes $h_{t-1}$ and $x_t$ as input and generates a number between 0 and 1 for each number in the previous cell state ($C_{t-1}$). If this value is 1, the LSTM preserves the corresponding information completely and if it is 0 then the LSTM discards the information completely. The working procedure of this forget gate ($f_t$) is represented by Eqn. 2.3. In the following step, the LSTM

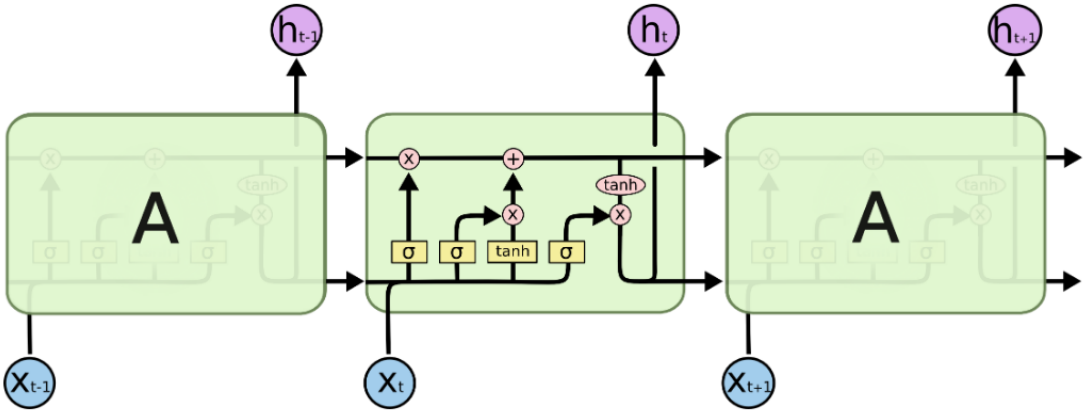



Figure 2.2: Long Short-Term Memory (source: Olah [47]).

makes a decision over the importance of the new information. This is done in two steps. In the first step, another sigmoid layer named the "input gate" decides which value to update. In the second step, a tanh layer produces a vector of new candidate cell state values ($\tilde{C}_t$) (Eqn. 2.4 and 2.5). Then the previous cell state $C_{t-1}$ is updated to $C_t$. To do so, $C_{t-1}$ is multiplied with the forget gate value $f_t$ to remove the unnecessary information from the past. Then, the input gate values are multiplied with the new candidate cell state values to store the important portion of the new input values. Later, these two values are summed to generate the new cell state memory (Eqn. 2.6). The last gating mechanism that the LSTM uses is the output gate. It decides the next hidden state ($h_t$) for the time step. $h_t$ propagates the information of the previous inputs. For this, the previous hidden state ($h_{t-1}$) and the current input ($x_t$) are passed to a sigmoid function (Eqn. 2.7). Then, the new cell state value is passed to a tanh function and multiplied with the sigmoid output to decide which information the hidden state would propagate to the next time step. This is also the hidden state ($h_t$) at the current time step $t$. Eqns. 2.3-2.8 are from Olah [47].

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2.3}$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2.4}$$

$$\tilde{C}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \tag{2.5}$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{2.6}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{2.7}$$

$$h_t = o_t * \tanh(C_t) \tag{2.8}$$

### Gated Recurrent Unit

The gated recurrent unit (GRU) is a variant of LSTM that couples input and forget gates and thus reduces computation costs to some extent. Unlike the LSTM which separately decides which information to preserve and which to get rid of, GRU makes both of these decisions at one time. To do so, it introduces an update gate. Again, it merges the cell state and the hidden state and introduces the reset gate. The update gate decides which new information to add and which information from the past to forget. The reset gate decides how much information from the past to forget. The working procedure of GRU is described by Eqns. 2.9-2.12 [47].

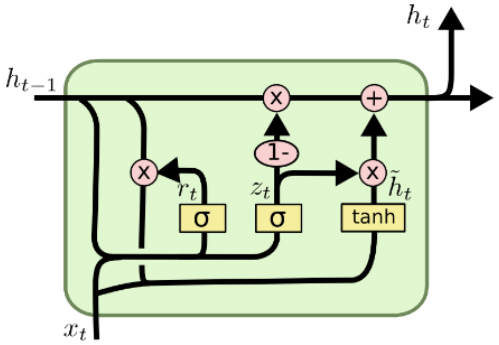



Figure 2.3: Gated recurrent unit (source: Olah [47]).

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t]) \tag{2.9}$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t]) \tag{2.10}$$

$$\tilde{h_t} = \tanh(W \cdot [r_t * h_{t-1}, x_t]) \tag{2.11}$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h_t} \tag{2.12}$$

### Bi-directional Recurrent Neural Networks

Bidirectional recurrent neural units are extensions of traditional recurrent units. Schuster and Paliwal [54] introduced BRNNs to provide the recurrent neural units with information from both past and future time steps while working with long sequential data [52]. These models train two recurrent units rather than one on the same input sequence. One unit works over the input sequence in the forward direction (from first to last). The other unit works with the very same input but in the reverse order. This approach provides the network with additional context as the information, in this case, comes from both the past and future time steps (forward and backward directions) and thus makes the convergence of the model faster and with a better

representation of the whole sequence [12]. Moreover, in the case of longer sequences, a simple sequential recurrent unit may lose some information from the distant past. This approach tends to prevent this situation as the sequence is processed from both directions and any information which appears at the very beginning for one module appears as the latest information to the other module. Finally, these models concatenate the hidden states of each RNN for corresponding time steps. Thus the output layer of this type of model gets information from the past (backwards) as well as the future (forward) states simultaneously [29]. For this bi-directional recurrent model, any kind of recurrent neural network can be used like simple RNN, LSTM or GRU. The procedure will be the same for all the variants. Figure 2.4 demonstrates the working procedure of a simple Bi-directional LSTM architecture.

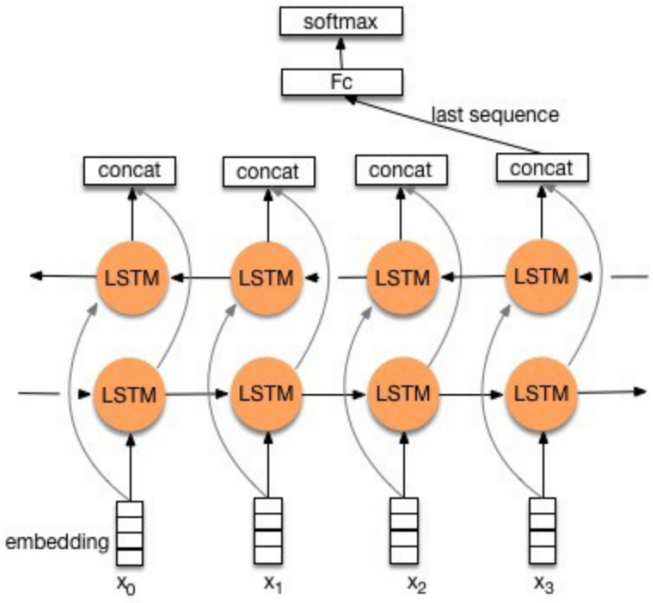



Figure 2.4: Bi-directional long short-term memory (source: Lee [32]).

### 2.2.2 Neural Network Based Word Embedding Models

Deep learning models are very capable of learning the inherent structures of the data and making the final decision from the data based on the given task. Still, they can't work with raw string data, rather these models require input in the form of numbers. Word embedding is the technique to map every word in the vocabulary to a numeric representation, more specifically a specific vector representation. The simplest way of mapping words to vector forms is the one-hot encoding technique. In one-hot encoding, the vector representation of a word is a binary vector where all the positions apart from only one are zero values and the remaining

position contains value 1. For example, if a vocabulary contains only three words: ['Natural', 'Language', 'Processing'], the one-hot representation of these words will be 'Natural': [1,0,0], 'Language': [0,1,0] and 'Processing': [0,0,1]. This technique has two drawbacks. Firstly, the dimension of the vector increases as the vocabulary size expands. It increases the computational complexity of the following models working with any large vocabulary. Secondly, this representation can't preserve any semantics of the words. To provide a vocabulary size-independent, semantic information preserving and fixed-sized vector representation, a few methods have been proposed in recent times. The following section discusses various techniques used for generating word embeddings.

### Neural Language Model

In 2003, Bengio et al. [3] utilized a feed forward neural network in order to learn a distributed representation of words. Given a training sentence, this model tries to learn the exponential number of neighbour sentences which are semantically similar. During the training phase, this model learns the probability function for word sequences along with the distributed representations of the words in the corpus. The architecture of neural network language model (NNLM) has four components: (i) the input layer, (ii) the projection layer, (iii) the hidden layer and (iv) the output layer.

This model follows the concept of auto-encoder (AE) in terms of working mechanism. The input-hidden layer portion of the NNLM works as the encoder portion of the auto-encoder whereas, the hidden-output layer portion of the model acts as the decoder portion. The input-hidden layer portion encodes the higher-dimensional one hot representation of the words into a lower dimensional vector representation. Then the following hidden-output layer portion maps this lower-dimensional vector representation to its original higher-dimensional representation. As a language model, NNLM takes $N$ previous words as input and tries to predict the next word in the sequence. The input words are fed to the model in the form of one-hot encoding. Similarly, the final output at the output layer is also one-hot encoding of the desired word at that end. The one-hot value from the input layer is projected to the projection layer $P$ with $N \times D$ computational complexity, where $D$ is the projection layer dimension. The following hidden layer computes the probability distribution of all the words. With an $H$ dimensional hidden layer, the computational complexity for mapping from projection to hidden layer is $N \times D \times H$. Finally, the $H$ dimensional hidden representation is mapped to the $V$ dimensional output layer, where $V$ represents the vocabulary size.

**Word2Vec and Distributed Representations of Words and Phrases and their Compositionality**

Mikolov et al. [42] proposed a predictive model (Word2Vec) for generating fixed-sized vector representations of words. This model is applicable for co-occurrence data with respect to a fixed sized context window. The context window is made of one target word and its surrounding context words. Word2Vec has two versions based on the training method:

1. Skip-Gram (SG): For a given target word the model is trained to predict the context words.

2. Continious Bag-of-Words (CBOW): Given a set of context words, based on the context window size, this model is trained to predict the target word.

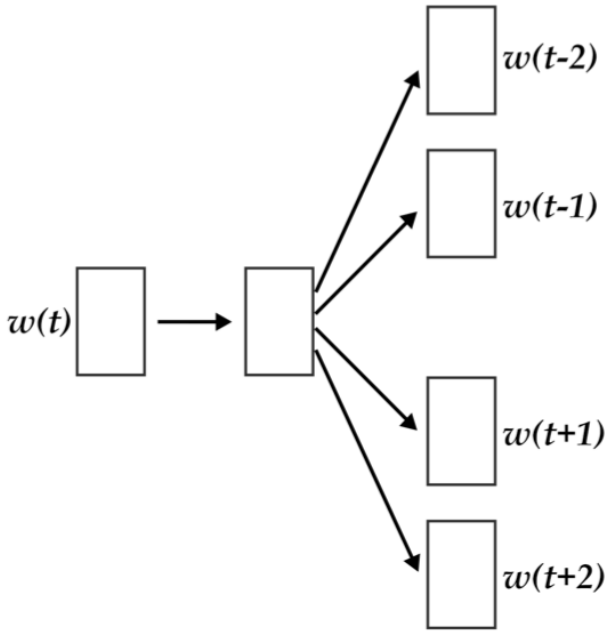



Figure 2.5: Word2Vec: Skip Gram Model (source: Mikolov et al. [42]).

Both of these models work on the principle of AE and try to map the vocabulary-size-dependent one-hot encoding of the words to a fixed sized lower dimensional vector representation preserving semantic information. To make it precise, every component of this vector is a float value, as floating point numbers manage precision in a better way. This shallow model is trained on a 1.6 Billion word corpus and the resulting word representation captures better semantic and syntactic information compared to other existing models at that time. They also claim that their model can capture the multiple degree similarity between the words as well as

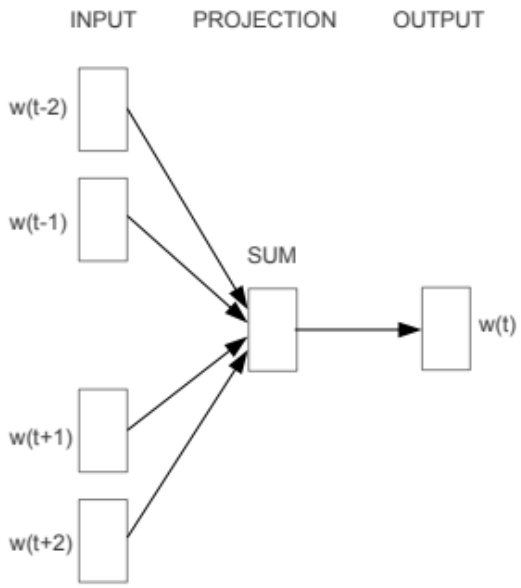



Figure 2.6: Word2Vec: Continuous Bag of Words Model (source: Mikolov et al. [42]).

just making the similar words much closer. For example, nouns can have similar word endings and we can find those words that have similar endings if we search for similar words in the original vector space. The concept of this model is based on Bengio's NNLM [3] model. The major difference is that both of the Word2Vec models omit the hidden layer unit of NNLM and establish a connection between the projection and output layers directly (Figure 2.5 and Figure 2.6). To maintain uniformity for all the words in the input layer this model takes all the words as one-hot encodings. And to calculate a distribution of probability over all the words in the corpus there is a one-hot form in the output layer as well. For both the cases (SG and CBOW), the model attempts to maximize the log probability of the output word for a given specific word or set of specific words (Eqn. 2.13) [43]. In order to calculate this probability distribution, they use a *softmax* layer at the output (Eqn. 2.14) [43].

$$J = \max(\sum_{t=1}^{C} \sum_{k=-n,k\neq 0}^{n} \log P(w_t|w_{t+k})) \tag{2.13}$$

$$p(w_{t+k}|w_t) = \frac{e^{(v'_{w_t})^\intercal v_{w_{t+k}}}}{\sum_{w\in V} e^{(v'_{w_t})^\intercal v_{w_{t+k}}}} \tag{2.14}$$

After both of these Word2Vec models are trained, it is possible to remove the decoder portion (the hidden-output layer portion) and then this model can generate low dimensional vector representation of the words. For CBOW this vector representation is for the context words and for the target words in the case of the SG model. For both of these models with

a larger context window size, these models come with higher accuracy, but the computational time is compromised. The overall computational complexity of this model is:

$$Q = N \times D + D \times log_2 V \tag{2.15}$$

However, this models' training time complexity utilizing simple *softmax* is not suitable for practical implementation as the cost of computing $\nabla\ log p(w_t|w_{t+k}) \propto |V|$. To solve this issue, later the authors came up with a solution to speed up the training process of Word2Vec [43]. The most important techniques are: i) Hierarchical Softmax and ii) Negative sampling [43].

**FastText**

Fasttext [4] follows the same architecture and concept of the skip-gram Word2Vec model with a small change. Unlike skip-gram, this model doesn't work with the vocabulary words, rather adds n-grams of subwords to the vocabulary. For example, the tokens generated for the word "*apple*" are *<ap, app, ppl, ple, le>*, *apple*, where the less than and greater than signs represent the beginning and ending of the word, respectively. Though this model improves the accuracy and gives better semantic and syntactic representation of the words, because every word is split into a bag of character n-grams, the vocabulary size grows larger and a separate dictionary function has to be maintained to find out the specific character n-gram. To adjust to the use of subwords the objective function of the skip-gram model is also modified. The objective function for FastText maximizes the sum of the vector representations of the words as well as the subwords (Eqn. 2.16) [4].

$$j = \sum_{t=1}^{C} \sum_{k=-n}^{n} \sum_{w \in G_{w_t}} \log p(w_{t+k}|w) \tag{2.16}$$

FastText does not only improve the quality of the word representations, but also has the ability to generate vector representations for the out of vocabulary words as this model has the vector representations for the subwords along with the vocabulary words. In case of any out of vocabulary words, this model adds the vector representations of all possible subwords to generate the vector representation.

**BioWordVec**

In 2017, Xhang et al. [65] trained the FastText model for biomedical natural language processing (BioNLP) tasks. Their model generates the word embeddings for the words found in biomedical texts. They trained their model on data from two sources: domain knowledge provided by Medical Subject Heading (MeSH) terms and biomedical literature found in PubMed,

a corpus of more than 25 million journal article abstracts. For this task they first constructed a MeSH term graph from the MeSH RDF data. They produced a number of MeSH term sequences utilizing a random sampling strategy. After that, they trained the FastText model over this data to teach the model text sequences and MeSH term sequences. The work-flow of BioWordVec training is presented in Figure 2.7.

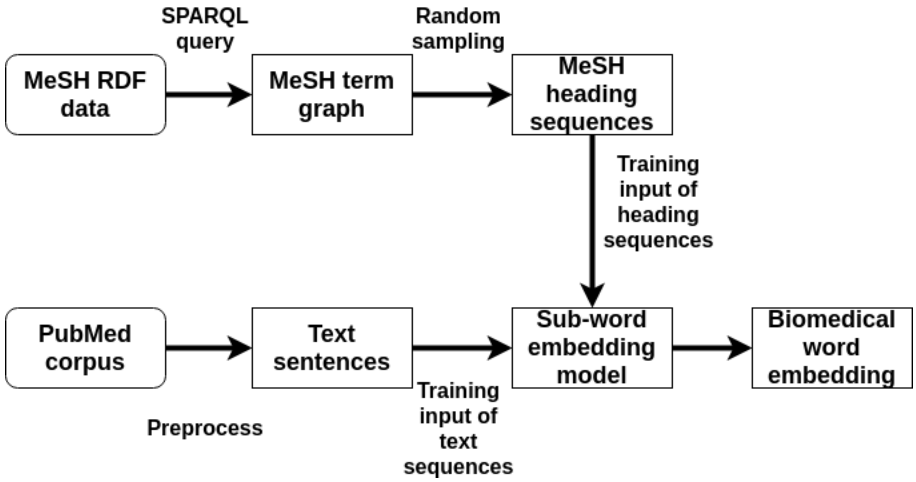



Figure 2.7: Schematic of learning word embedding based on PubMed literature and MeSH (source: Xhang et al. [65]).

### 2.2.3 Sentence Embedding

Word embedding techniques allow the deep learning models to work with the vector form of the words and these vectors to some extent preserve semantic relatedness. However, for the case of expressing information, having only words is not sufficient, rather, we need sentences. One way to deal with this issue is to take the word vectors of any particular sentence and make some simple mathematical operations like average them (a normalized sum) to generate the sentence representation. However, this is not somehow the best way to preserve the semantic meaning of the sentence. To understand the proper meaning of a sentence, information from the context is also necessary. Keeping this idea in mind, a few sentence embedding techniques have been proposed in recent times. Moreover, the length of the sentences are variable in nature where deep learning models have the limitation that they work with fixed length data. This section discusses some unsupervised sentence embedding techniques that try to generate fixed-length vectors for sentences with some information about the context.

### Skip-Thought Vectors

In 2015, Kiros et al. [31] proposed an unsupervised sentence embedding model that tries to capture contextual information to generate distributed vector representations of sentences. This model is designed based on the skip-gram idea [42]. Skip-gram tries to generate the vector representations of the context words given a target word. Here in the case of skip-thought, the model tries to generate the distributed vector representations of the context sentences given a target sentence.

The skip-thought model follows the principle of the encoder-decoder architecture. The encoder portion tries to map the words of a sentence to a sentence vector, where the decoder portion generates the sentence vector for the surrounding context. For the encoder-decoder model, they explored a a variety of deep learning models like RNN-RNN, ConvNet-RNN, LSTM-LSTM, etc. and got the best result for the GRU-GRU architecture.

The GRU-GRU model can be decomposed into three portions: i) the encoder, ii) the decoder, and iii) the objective function. For a sequence of $N$ words $(w_i^1, \ldots, w_i^N)$ in a sentence $S_i$, at each time step $t$, the encoder generates a vector representation of the word sequence $(w_i^1, \ldots, w_i^t)$ up till that time step $(t)$ which is basically the hidden state representation $(h_i^t)$ of the GRU unit. So, the hidden state of the GRU after it processes all the words in the sentence is the representation of the sentence $(h_i^N)$. They designed the decoder like a neural language model conditioning on the output of the encoder $(h_i)$ for sentence $s_i$. The decoder portion adds additional biases on the update and reset gates and the hidden state of the GRU. The other operations are similar to those of the encoder. This architecture uses two separate decoders with the same architecture but different parameter settings. One decoder is assigned to generate the sentence vector of the next sentence $(s_{i+1})$ in the context after the target sentence $(s_i)$, while the other decoder is used to do the same task for the previous sentence $(s_{i-1})$ in the context window.

### QuickThought

In 2018, Logeswaran and Lee [37] proposed another framework which can learn the vector representation of sentence from unlabeled data. The idea is similar to the skip-thought [31] model in the sense that it also tries to predict the context sentence given a target sentence, but not in the way of sentence generation rather by means of classification technique. The idea behind this model is quite similar to the idea of negative sampling [43, 57]. For a given target sentence, QuickThought tries to classify the appropriate and contrastive context sentences. To achieve this goal, this model replaces the generative objective function of skip-thought with a discriminative approximation function. Because of this approach this model is faster in terms of training time, and at the same time, produced the state of the art performance at that time.

Quick-Thought utilizes the target sentence's meaning to predict the appropriate context sentences with similar meaning. Here, the term "similar meaning" refers to the similarity between the vector representations of the sentences. This approach helps the model to preserve better semantics in the sentence vectors. Given few candidate context and the target sentences, the encoder portion encodes the sentences at first. Then from the set of candidate sentences Quick-Thought selects the correct one.

Suppose, $f$ and $g$ are two functions ($f$ and $g$ may or may not share the same parameter set) responsible for the encoding of the sentences. For a target sentence $s$, let $S_{context}$ be the set of true context sentences, and $S_{candidate}$ be the set of candidate sentences for a true context sentence $s_{context}$ where a few candidates are contrastive. So, for a given candidate sentence $s_{candidate} \in S_{candidate}$ the probability to be a true context sentence is Eqn. 2.17 [37]:

$$p(s_{candidate}|s, S_{candidate}) = \frac{\exp[c(f(s), g(s_{candidate}))]}{\sum_{s' \in S_{candidate}} \exp[c(f(s), g(s'))]} \tag{2.17}$$

where, $c$ denotes the classification score function which is a simple inner dot product: $c(u, v) = u^T v$. The objective function of the model tries to maximize the true context sentence's probability for a given target sentence (Eqn. 2.18) [37].

$$\sum_{s \in D} \sum_{s_{context} \in S_{context}} \log p(s_{context}|s, S_{candidate}) \tag{2.18}$$

After the model training is done, this model disposes the classifier portion and only the encoder portion is used for generating the sentence vectors. For any sentence $s_i$ the final sentence vector representation is $[f(s), g(s)]$ (concatenation of sentence vectors generated from both of the encoder functions).

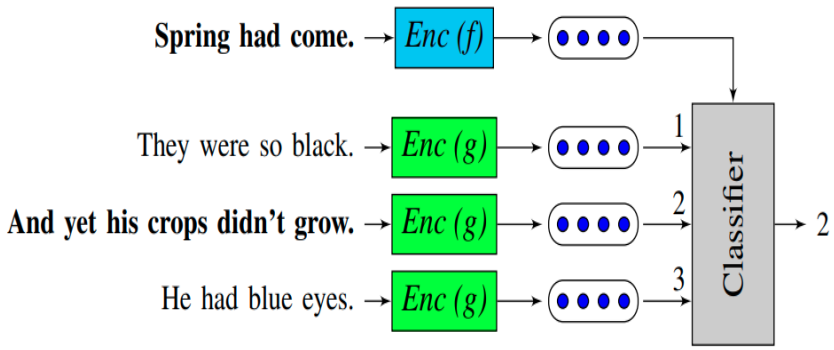



Figure 2.8: Quick Thought Model (source: Logeswaran and Lee [37]).

### Sent2Vec

Sent2Vec [48], a general purpose sentence embedding model, is designed as an extension of the CBOW model [42] with the intention to generate sentence embeddings rather than word embeddings. This model composes the vector representation of the sentences using the vector representations of the words (with n-gram embedding). At the same time, this model does the training of the word embeddings as well. This model can be described as an optimization problem like Eqn. 2.19 [48]:

$$\min_{U,V} \sum_{S \in C} f_S(UV\iota_S) \tag{2.19}$$

where $\mathcal{V}$ denotes the vocabulary, $U \in \mathbb{R}^{\mathcal{V} \times h}$ and $V \in \mathbb{R}^{h \times |\mathcal{V}|}$ are the two parameter matrices. Columns in matrix $V$ and $U$ indicate the vector representation of the context words and target words respectively. $\iota_S \in \{0,1\}^{|\mathcal{V}|}$ is a vector where a position is "1" only if that word is present in the sentence $S$. To serve the purpose of sentence embedding, $S$ (context window) can be either a portion of the document or the entire document.

Sent2Vec tries to learn both the context ($v_w$) as well as the target embedding ($u_w$) for every word $w \in \mathcal{V}$. Later averaging context embedding vectors for all the words in the sentence the final sentence embedding ($v_S$) is generated. This model incorporates not only the unigram words but also n-grams present in each sentence. The final formula for the sentence embedding for any sentence $S$ is defined as [48]:

$$v_S = \frac{1}{|R(S)|} V\iota_{R(S)} = \frac{1}{|R(S)|} \sum_{w \in R(S)} v_w \tag{2.20}$$

where $R(S)$ denotes the set of all possible n-grams plus the unigrams for any sentence $S$. Later this model uses negative sampling [43] along with the binary loss function $\ell : x \rightarrow \log(1 + e^{-x})$ and formulates the objective function as [48]:

$$\min_{U,V} \sum_{S \in C} \sum_{w_t \in S} (\ell(u_{w_t}^T v_{S \setminus \{w_t\}})) + \sum_{w' \in N_{w_t}} \ell(-u_{w'_t}^T v_{S \setminus \{w_t\}}) \tag{2.21}$$

where $N_{w_t}$ corresponds to the negative sampled words for the word $w_t \in S$. The negative samples are selected from a multinomial distribution. In this distribution, every word $w_t$ is assigned with a probability $q_n(w) = \sqrt{f_w}/(\sum_{w_i \in \mathcal{V}} \sqrt{f_{w_i}})$. Here, $f_w$ denotes word $w_t$'s normalized frequency with respect to the corpus. For the purpose of subsampling, any word $w_t$ is discarded with a probability $1 - q_p(w_t)$. $q_p(w_t)$ is defined as [48]:

$$q_p(w_t) = \min\{1, \sqrt{t/f_{w_t} + t/f_w}\} \tag{2.22}$$

Table 2.1: Corpora for training BioSentVec embeddings (source: Chen et al. [14]).

| Corpus | Documents | Sentences | Tokens |
|---|---|---|---|
| PubMed | 28,714,373 | 181,634,210 | 4,354,171,148 |
| MIMIC-III | 2,083,180 | 41,674,775 | 539,006,967 |

This subsampling method prevents the model from being biased by the most frequent words in the corpus. So, the final objective function of the model is [48]:

$$\min_{U,V} \sum_{S \in C} \sum_{w_t \in S} (q_p(w_t) \ell(u_{w_t}^T v_{S \setminus \{w_t\}})) + |N_{w_t}| \sum_{w' \in N_{w_t}} q_n(w') \ell(-u_{w'_t}^T v_{S \setminus \{w_t\}}) \tag{2.23}$$

Along with providing state of the art performance, this model is very simple and computationally cost efficient. Once the model is trained, for any sentence $S$, it requires only $|R(S) \times h|$ floating point operations for the n-gram. The computational complexity of this model is only $O(1)$ vector operations per word. Moreover, because of its straight-forward nature, it supports parallel training using parallel stochastic gradient descent.

#### BioSentVec

Chen et al. [14] trained the Sent2Vec model [48] over 30 million documents combining articles from PubMed and clinical notes in the MIMIC-III clinical dataset. They evaluated their model on two different semantic similarity tasks: i) sentence similarity and ii) multi-label text classification. They claimed that their model captures better semantics at the sentence level compared to other alternatives in the biomedical domain and produces state of the art performance. For the training of their model, they built a corpora with 28,714,373 documents from PubMed and 2,083,180 documents from the MIMIC-III dataset.

Texts from both PubMed and MIMIC-III were split and tokenized using the NLTK framework. Then they trained the Sent2Vec model [48] over this dataset. The output sentence embeddings are 700-dimensional vectors. For their experiments they set the window size to 30 and utilize 10 negative examples for each word. They utilized the bi-gram along with the words in the sentence for model training. Table 2.1 gives some details about the corpora they made for training the BioSentVec.

### 2.2.4 Attention Mechanisms For Natural Language Processing Tasks

The basic principle behind the "attention" mechanism is that whenever a model tries to predict a single output word, it doesn't need to focus on the whole sentence, rather only on portions of the sentence where the corresponding information is concentrated. This process is similar to the

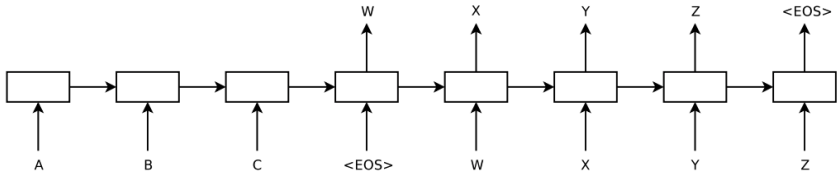



Figure 2.9: Sequence to sequence architecture (source: Brownlee [10]).

human visual system. While looking at any image, the human eye focuses on a certain portion of the image with higher resolution without completely ignoring the surrounding portions with lower resolution. Over time the focal point is adjusted to capture the information inherent in the image. The attention mechanism in NLP is similar to the non-local means algorithm [13] from the image processing domain. Recently, attention mechanisms have made their way into recurrent neural network architectures that are typically used in NLP. Different attention mechanisms and their application will be discussed in this section.

### Introduction to the Concept of Attention

Neural machine translation (NMT) is the technique of applying a neural network architecture for the machine translation task. NMT intends to build a single end-to-end neural network that can be trained to maximize the translation performance. In most of the cases, these models somehow follow the encoder-decoder architecture where the encoder portion generates a fixed length vector from the source sentence and the translation is generated by the decoder portion. For the encoder and decoder portions different neural network models can be used. For example, Sutskever et al. [58] used LSTM for the machine translation task. Figure 2.9 shows a general encoder-decoder architecture used as the backbone in most of the NMT tasks. The encoder reads the entire input sequence as a vector $x = \{x_1, x_2, \ldots, x_t\}$ and transforms it into a fixed-length vector $c$. Then, the decoder tries to predict the next word given all of the words generated before $\{y_1, y_2, \ldots, y_{t-1}\}$.

However, for the decoder to do language modeling based on just the encoded hidden state is not feasible because a single context vector cannot encode the entire sequence information if the sequence length is too long. That's where the attention mechanism came to the rescue because it allows the decoder to search through a source sentence during decoding a translation and calculates which portion to attend to more and vice versa.

In 2014, Badhanau et al. [2] introduced the idea of attention in the task of NMT, which defines how much importance a source word should get from the model while generating a target word by the decoder portion. In this case, the term "context" means the last hidden state value of the encoder plus the last hidden state of the decoder neural network unit at each time

step. A demonstration of the NMT with attention mechanism is portrayed in Figure 2.10.

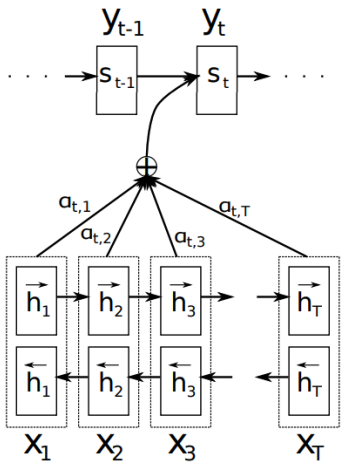



Figure 2.10: An illustration of the attention mechanism (indicated by $\oplus$ and the attention weights $\alpha_{i,j}$) to generate the target word from the source sentence (source: Badhanau et al. [2]).

This mechanism changes the context vector for each decoded word. This indicates that the model is giving variable importance to different portions of the source sentence while decoding different translated words. This process changes the joint probability from Eqn. 2.24 and becomes:

$$p(y_i|\{y_1, y_2, \ldots, y_{i-1}\}, c) = g\ (y_{i-1}, \tilde{h}_i, c_i) \tag{2.24}$$

where $\tilde{h}_t$ is the hidden state of the decoder RNN for time $i$. $\tilde{h}_t$ is computed by,

$$\tilde{h}_i = f\ (\tilde{h}_{i-1}, y_{i-1}, c_i) \tag{2.25}$$

The context $c_i$ now depends on the weighted sum of the sequence of $h_i$:

$$c_i = \sum_{j=1}^{T} \alpha_{ij} h_j \tag{2.26}$$

The weights $\alpha_{ij}$ are computed by

$$\alpha_{ij} = \frac{\exp\left(e_{ij}\right)}{\sum_{k=1}^{T} \exp\left(e_{ik}\right)} \tag{2.27}$$

where

$$e_{ij} = a\ (\tilde{h}_{i-1}, h_j) \tag{2.28}$$

Here, $e_{ij}$ reflects how well the output word at the $j^{th}$ position aligns with the input word at the $i^{th}$ position. This shallow neural network is trained simultaneously with the overall architecture of the NMT model. Here, Eqns. 2.24-2.28 are from Badhanau et al. [2].

### Inner Attention

In 2016, Liu et al. [36] proposed a sentence encoding model for the text entailment recognition (RTE) task where instead of applying the attention mechanism over source and target sentences, words in each individual sentence attend over themselves. They named this attention mechanism "Inner Attention". Their model works in two steps. In the first step, mean-pooling is applied over the word-level Bi-LSTM. This step generates one vector representation of the sentence. In the second step, their proposed inner attention mechanism is applied instead of mean-pooling to get another vector representation of the sentence. They conducted their experiments on the Stanford Natural Language Inference (SNLI) corpus, a corpus built for the RTE task.

The RTE task is to determine whether an hypothesis can be inferred from a given premise. This is a three class classification task. The three class labels are entailment, contradiction and neutral. A few samples of the dataset are displayed in Table 2.2

This model is a framework for the RTE task which omits the necessity of feature engineering or any kind of additional resources. The baseline of this model uses a Bi-LSTM to generate a vector representation for both the hypothesis and premise sentences. The architecture of the model consists of three modules: i) the sentence input module, ii) the sentence encoding module, and iii) the sentence matching module (Figure 2.11). The sentence encoding module works in two steps. Mean-pooling at the first stage gives a rough insight about the information any particular sentence wants to express. The inner attention mechanism in the second stage fine-tunes the quality of the representation gained from the first step. The idea of the inner attention mechanism originates from the observation that while reading a sentence, a human forms a rough intuition about the importance of different portions of the sentence based on previous portions. The formulation of the inner attention mechanism is as follows (Eqns. 2.29-2.31) [36]:

Table 2.2: Examples of three types of labels in RTE (source: Liu et al. [36]).

| **Premise** | The boy is running through a grassy area. | **Label** |
|---|---|---|
| **Hypothesis** | The boy is in his room. | Contradiction |
| | A boy is running outside. | Entailment |
| | The boy is in a park. | Neutral |

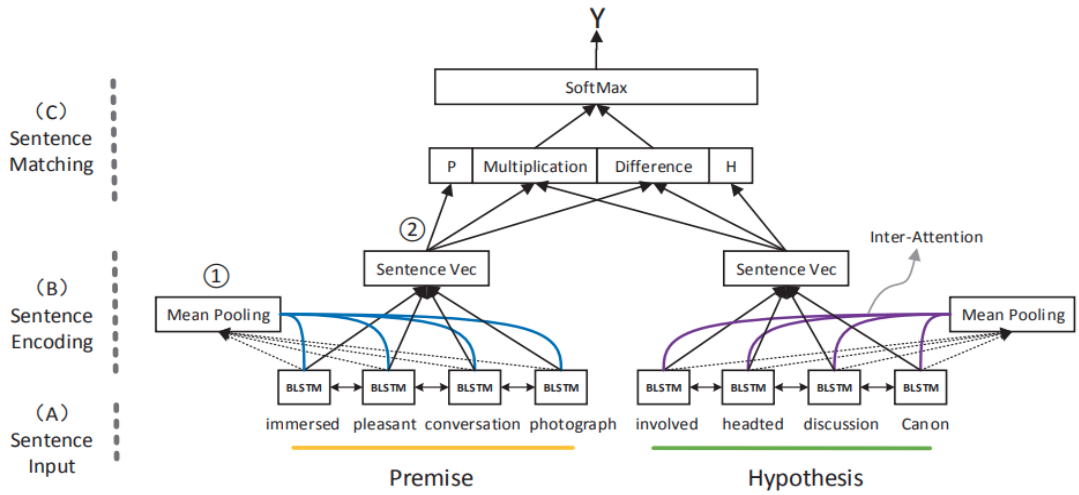



Figure 2.11: Inner-Attention mechanism over Bidirectional LSTM (source: Liu et al. [36]).

$$M = \tanh(W^y Y + W^h R_{mean} \otimes e_L) \tag{2.29}$$

$$\alpha = softmax(w^T M) \tag{2.30}$$

$$R_{attention} = Y\alpha^T \tag{2.31}$$

Here, $Y$ is the matrix containing the output vectors of the Bi-LSTM, $R_{mean}$ denotes the output of the mean-pooling layer, $R_{attention}$ represents the attention ($\alpha$) weighted vector representation of the sentence.

Furthermore, the authors developed a technique to remove the common sentences from the hypothesis and premise sentences which boosts the performance of the model further. This model beat the state-of-the-art model of that time by around 2 percentage points.

### 2.2.5 Hierarchical Attention Network

In 2016, Yang et al. [63] proposed a hierarchical attention architecture to classify documents. This approach utilizes the attention mechanism at two levels. First, the attention mechanism [2] is applied over the words in a particular sentence to generate the sentence embedding. Then, the attention mechanism is applied over the sentence representations to determine the importance of each sentence in the document. Finally with the attention weighted sentence representation, the representation for the document is generated.

This hierarchical attention network consists of four different parts 2.12. In the first part, Bi-GRU is applied over the word representations of a single sentence where at each time step one

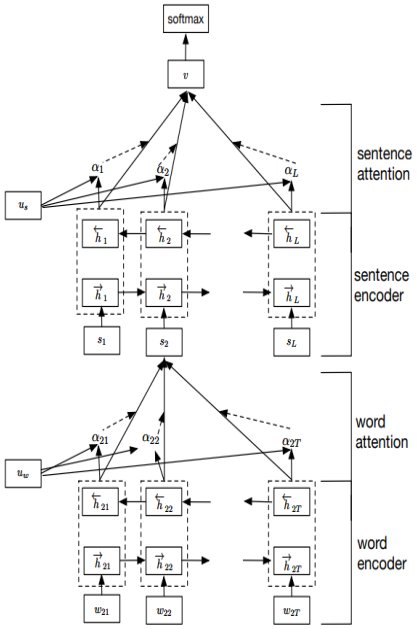



Figure 2.12: Hierarchical attention network (source: Yang et al. [63])

word representation is fed to the GRU. By concatenating the corresponding hidden state representations, the word representation is generated. In the second part, the attention mechanism is applied over these word representations to generate the sentence representation as follows (Eqns. 2.32-2.34) [63]:

$$u_{it} = \tanh(W_w h_{it} + b_w) \tag{2.32}$$

$$\alpha_{it} = \frac{\exp(u_{it}^T u_w)}{\sum_t \exp(u_{it}^T u_w)} \tag{2.33}$$

$$s_i = \sum_t \alpha_{it} h_{it} \tag{2.34}$$

where $h_{it}$ represents the hidden state representation for the $t^{th}$ word in the $i^{th}$ sentence. $h_{it}$ is fed to a single layer MLP to generate the hidden layer representation of $h_{it}$ ($u_{it}$). The importance of the word $w_{it}$ ($\alpha_{it}$) is measured against a word level context vector $u_w$. Finally, *softmax* is applied to get the normalized attention value. The sentence vector representation is computed by adding the attention weighted word representation of the words in the sentence. In the next part of the model, the sentence vectors are treated by Bi-GRU as was done with the word embeddings and a sentence level annotation is produced. After that, the attention mechanism (as stated earlier) is applied over these sentence annotations to generate the vector representation

of the whole document. This time another context vector ($u_s$) at the sentence level is used to compute the normalized attention value of each sentence in the document. Both of the word level and sentence level context vectors are initialized randomly and trained throughout the training process.

### Structured Self-Attention

Lin et al. [35] proposed a new mechanism for generating sentence embedding with a new type of attention procedure called structured self-attention. Unlike the other methods which generate a vector representation of the sentence, the structured self-attention mechanism produces a 2-D matrix representation for particular sentences. Each row of this matrix depicts the importance put on different portions of each of these sentences. This model is very helpful when there is only one sentence rather than a sentence pair to work with like the sentiment classification task. This model allows the down-stream task to put different attention values on different parts of the sentence.

This sentence embedding model consists of two modules: i) a Bi-LSTM and ii) the self-attention mechanism. This self-attention mechanism outputs a summation of the dot product of the weight vectors and the hidden states of the LSTM network to generate the final sentence embedding matrix. Figure 2.13 shows the architecture of the model for the sentiment classification task.

For a sentence $S$ with $n$ words, $S$ can be presented as a sequence of corresponding $d$-dimensional word embeddings ($w_i$).

$$S = (W_1, \ldots w_n) \tag{2.35}$$

Thus, $S$ is a 2-D matrix where $S \in \mathbb{R}^{n \times d}$. To maintain some dependency between independent adjacent words a Bi-LSTM is operated over this matrix $S$ and the hidden dimension at each time step from both directions are concatenated to obtain the final hidden state $h_t \in \mathbb{R}^{2u}$ where the hidden dimension of the LSTM unit is $u$. Combining all the $h_t$ at different time steps the matrix $H \in \mathbb{R}^{n \times 2u}$ is formed. The self-attention mechanism takes this $H$ and outputs a attention vector $A \in \mathbb{R}^n$ as follows:

$$A = softmax(w_{s2} \tanh(w_{s1} H^T)) \tag{2.36}$$

Here, $w_{s2}$ is a vector of dimension $d_a$ and $w_{s1}$ is a matrix of dimension $\mathbb{R}^{d_a \times 2u}$. The value of the hyper-parameter $d_a$ can be picked arbitrarily. The final vector representation of the sentence

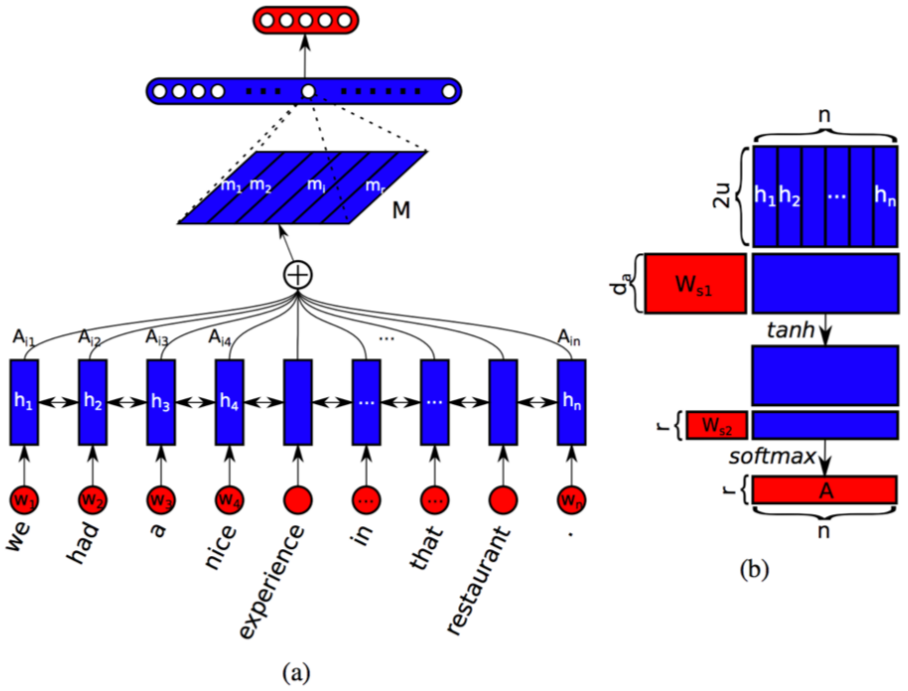



Figure 2.13: Self-attention mechanism for sentiment classification task (source: Lin et al. [35]).

($m \in \mathbb{R}^{2u}$) can be achieved by:

$$M = AH \tag{2.37}$$

This vector representation of the matrix $M$ focuses on a specific word or phrase of a sentence. However, in case of long compound sentences, there may be multiple places where higher importance should be given. To resolve this issue, the authors came with a multi-hop idea for attention mechanism. In the multi-hop technique, for $r$ different important parts of the sentence, the vector $w_{s2}$ is extended to a $r \in d_a$ matrix ( $W_{s2}$). Thus the attention vector $a$ becomes attention matrix $A \in \mathbb{R}^{r \times n}$ and the sentence embedding vector $m$ becomes sentence embedding matrix $M \in \mathbb{R}^{r \in 2u}$ where $M = AH$. However, the main ideology of this model may be compromised if the attention mechanism always provides similar summation weights for all the $r$ hops [35]. To prevent this issue, the authors added one penalization term $P$ where:

$$P = ||(AA^T - I)||_F^2 \tag{2.38}$$

Here, $P$ depicts the score redundancy, at the same time, is computationally faster than the KL-divergence. Eqns. 2.35-2.38 are from Lin et al. [35].

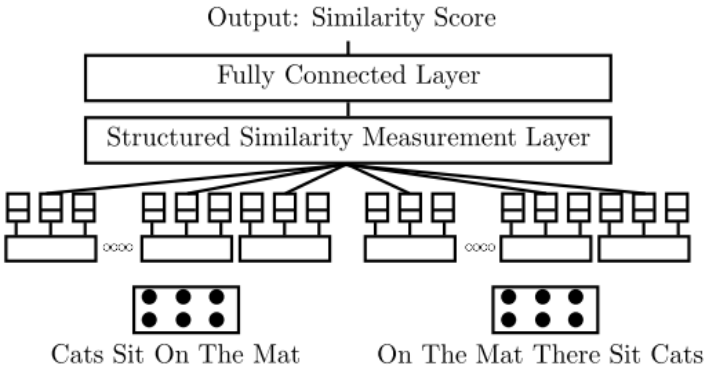



Figure 2.14: MP-CNN architecture for semantic relatedness task (source: He et al. [21]).

### 2.2.6 Attention-Based Multi-Perspective Convolutional Neural Network

In 2015, He et al. [21] proposed a semantic similarity measurement model for sentences using multiple convolution neural networks (CNN) with different convolution operations using different window size and pooling types. They named their model multi-perspective CNN (MP-CNN). The semantic similarity task can be viewed as a two class classification problem. Given two sentences ($S_1$ and $S_2$), this model returns "1" if these two sentences are semantically similar and "0" otherwise. For the similarity measurement they incorporated three types of distance functions: cosine, Euclidean and element-wise differences. They demonstrated state-of-the art performance on two SemEval semantic relatedness tasks. They emphasize the fact that this model doesn't need any additional parser or part-of-speech tagger to achieve this result. However, this model is very complex in nature and requires a lot of computational time.

The MP-CNN model has two major components: a sentence model and a similarity measurement layer. For conducting operations over two different sentences, this model uses two CNNs in parallel. For these parallel CNNs, MP-CNN follows the Siamese structure [9]. In the end, the output features from these two CNNs are joined by the similarity measurement layer followed by a fully connected layer for final class prediction. Figure 2.14 gives an overview of the MP-CNN model.

In 2016, the authors extended this model [22]. For the extended version of MP-CNN, the CNNs are fed with attention-based word embeddings instead of direct word embeddings. They used the Paragram-Paraphrase word embedding as the initial word embedding and later applied some attention mechanism over this Paragram-Paraphrase word embedding to generate the attention-based word embedding. Figure 2.15 shows the overall architecture of the MP-CNN model with the attention-based input interaction layer. For any sentence pair ($\ell_0$ and $\ell_1$), each sentence is represented by a matrix $S^i \in \mathbb{R}^{\ell_i \times d}$ ($i \in \{0, 1\}$) where $\ell_i$ represents the length of the sentence and $d$ denotes the dimension of the word embedding. $S^i[a]$ represents the embedding

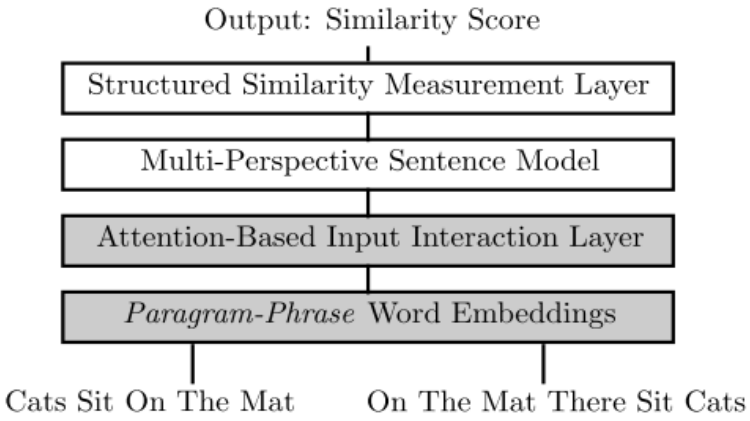



Figure 2.15: MP-CNN model with attention-based input interaction layer (source: He et al. [22]).

of the word at the $a^{th}$ position in the sentence. They defined an attention matrix $D \in \mathbb{R}^{\ell_0 \times \ell_1}$ where $D[a][b] = cosine(S^0[a], S^1[b])$. Given the matrix $D$, the attention weight vector $A^i \in \mathbb{R}^{l_i}$ is generated where each element corresponds to the attention-based relevance score of that corresponding word. The formulation of vector $A^i$ is as follows (Eqn. 2.39) [22]:

$$E^0[a] = \sum_b D[a][b]E^1[b] = \sum_a D[a][b]A^i = softmax(E^i) \tag{2.39}$$

Then the final attention based word embedding is generated by concatenating the original word vector and the point-wise multiplication of the original word vector with corresponding attention-based relevance score ($concat(S^i[a], A^i[a]) \odot S^i[a]$). Though this model gives state-of-the-art result, this performance comes at a cost of high computational operations as a lot of CNNs are being used for the feature generation task.

## 2.3 Semantic Similarity Measurement

Many problems in understanding natural language can be formulated as textual semantic similarity measurement between pairs of text spans. The text spans can be a single sentence up to a whole document. Similarity between two pieces of text can be measured in two ways: lexically and semantically. Lexical similarity measurement techniques look for common words between different pieces of text, whereas semantic similarity measurement techniques try to infer the semantic relationship between the texts. These semantic similarity measurement techniques are very important for NLP tasks such as the natural language inference task, where given a hypothesis sentence and a candidate premise sentence the technique tries to figure out whether

the premise is inferred from the hypothesis or not. Usually, this is formulated as a classification problem. Other examples of this semantic relatedness measurement task are paraphrase detection, information retrieval, summarization, etc. In case of information retrieval task, semantic relatedness measurement problem is formulated as a ranking problem where, given a query candidate documents are ranked according to the semantic relatedness between the query and each document. In case of question answering systems, semantic similarity measurement task is converted as into a classification task where given two questions, the models try to figure out whether two sentences are similar or not.

Much research has been conducted for modeling the inherent semantic relatedness between text spans. Traditional methods like tf-idf or bag-of-words have been implemented in a lot of work for measuring the semantic similarity between texts. However, these traditional models have a severe limitation. They consider two text spans similar only when both of the compared texts contain similar terms. However, two texts can be semantically similar even if they don't share similar terms. Furthermore, word positioning plays a vital role in modeling the semantics of a text. These traditional models completely ignore these issues.

Deep learning based approaches have solved this problem as they have the ability to learn the latent property of a text through the training step. These models represent the sentences as vectors and then compute the vector distance between two sentences to measure the semantic relatedness between them. In 2014, Shen et al. [56] applied a convolutional neural network with different pooling operations for the textual semantic similarity measurement task at the sentence level. Using a sliding window, this model splits the text into n-grams of words. For each window, the three-gram characters are extracted and this is done for every word. Then, a boolean vector of three-gram characters are used to represent the words and these word vectors are concatenated in a following step to be the feature vector of the sliding window. A convolution operation with a max-pooling layer is applied over each sliding window to generate feature vectors for the following feed-forward neural network which determines the semantic relatedness between two sentences. In 2018, Yao et al. [64] introduced a sentence encoder architecture using an LSTM for the semantic similarity measurement task. In the first stage, they introduced a new normalization algorithm to overcome the vanishing gradient problem and then utilized an inception module approach [59] to extract useful features from multiple dimensions. Finally, they used the cosine similarity metric to compute the semantic relatedness between two sentences. Shashavali et al. [55] investigated models with the cosine similarity computation incorporating both a sliding window and a weighted N-gram to generate vector representations for sentences with the intention to compute textual semantic relatedness. Mueller and Thyagarajan [46] utilized the siamese architecture [9] with an LSTM for this task.

However, these models give equal preference to all of the words in the sentences while

computing the semantic relatedness, whereas some words in a sentence should be given higher importance. For example, the word "not" can change the overall meaning of the whole sentence. In that case, this word should get higher importance. To deal with this issue, a few models have been introduced in recent times incorporating attention mechanisms for the semantic similarity measurement task. Yang et al. [36] introduced inner attention to compute the individual word's importance in a sentence and applied it with Bi-LSTM for the natural language inference classification task. For the same task, Lin et al. [35] introduced the structured self-attention mechanism where each sentence is represented by a matrix instead of a vector. Each row in that matrix represents different important portions of the sentence with the appropriate importance. Conneau et al. [15] incorporated the siamese architecture with these attention mechanisms and Bi-LSTM for the semantic relatedness measurement task.

# Chapter 3

# Methodology

This chapter gives an overview of the complete project from the data preprocessing to the citation linkage task. As little work has been done previously for the citation linkage task, a proper dataset for the training and testing of the models used for this task is not available. For this reason, we collected and preprocessed data for the training of the model. This dataset creation together with generation of word embeddings, sentence embeddings, and the final model design are the topics to be discussed in this chapter.

## 3.1 Dataset Creation

For the citation linkage task in the biomedical research article domain, only one dataset is available as per our knowledge. This dataset was created by Houngbo and Mercer [26]. Although in scientific research papers the citation span can be one or more sentences (or part of a sentence), for the above work, it is limited to a single sentence, so the citation linkage dataset is composed of sentence pairs. These sentence pairs were annotated by a domain expert for likelihood of being a citing-cited sentence pair on a scale of 0 (not likely) and 1 (lowest confidence score) to 5 (highest confidence score). This dataset is small. It contains only 3857 sentence pairs. This small amount of data is not sufficient enough for the training of deep learning models. Deep learning models have the ability to learn the inherent patterns of the data through the training process, but they need a lot of data to be accurately trained. Moreover, the dataset is highly imbalanced. Looking ahead to Section 3.2 we view the citation linkage task as a binary classification task, a view also suggested in [26]. Out of these 3857 samples only 81 are annotated with rating 4 or 5 which in our experiments are what we have considered positive samples and all others are negatively annotated. A model trained with this type of imbalanced dataset becomes biased.

To overcome these issues, we have developed a synthetic corpus of 68,898 sentence pairs

over three biomedical topics: cell biology, biochemistry, and chemical biology to train our deep learning models. Looking ahead again to Section 3.2, in this thesis we consider that a citing sentence refers to a cited sentence if they are semantically similar. In creating the dataset, we wanted to annotate the data without any human effort. For this reason, an unsupervised learning technique needs to be involved. The synthetic corpus has been annotated, not by humans, but rather by an unsupervised sentence embedding technique called Sent2Vec followed by a cosine calculation of the angle between the resulting sentence vectors as a measure of semantic similarity of the two sentences in each pair. Among these data, 45.89% samples are positive samples and the remaining are negative. We have used the corpus built by Houngbo and Mercer [25] for the validation and test purposes with a change in the scoring factor that was mentioned previously: scores 4 and 5 in this corpus are replaced with 1 (positive) and the remaining scores are replaced with 0 (negative).

Data must be collected for the creation of the synthetic corpus and as training data for Sent2Vec. Our data sources are the citation sentences from 2289 articles manually collected from the web and a set of 28,310 full-text articles from a wide spectrum of biomedical journals (first made available by The National Center for Biotechnology Information (PubMed Central) in 2009) that is used to train Sent2Vec. Upon inspection of these articles, we noticed that as these articles are from various biomedical genre, they differ in format because of genre and journal writing styles. For this and some other reasons that will be discussed below, the data needs to be cleaned before it can be used. Before discussing the data cleaning procedure in Section 3.1.3, we first turn to a description of the data and data sources and the creation of the synthetic corpus.

### 3.1.1 Sentence Embedding

In order to calculate the cosine of the angle between the sentence pair vectors in the synthetic corpus as a measure of semantic similarity of the two sentences in each pair, the sentences must first be embedded in a vector space. We have chosen Sent2Vec to perform this embedding. BioSentVec also provides sentence embeddings for biomedical sentences. However, upon inspection we noticed that few chemical names in our corpus are absent from their data. Furthermore, BioSentVec is trained with sentences from abstract section of the biomedical research papers only which are usually short in length. Where, sentences in our case are comparatively long in length and BioSentVec was not producing good sentence representations for them, That's why we trained Sent2Vec with our data instead of using pre-trained BioSentVec.

To generate good sentence embeddings, Sent2Vec must be trained on sentences taken from the same domain. Like other unsupervised sentence embedding models, Sent2Vec requires a

lot of data compared to supervised models. For this reason, 4,843,756 sentences from 28,310 research documents were collected. These documents are from 90 biomedical subdomains like cell biochemistry, biology, bioinformatics, biological chemistry, cell biology, etc.

After cleaning the sentences, the next step is to train an unsupervised sentence embedding model. This step is important for the annotation of the sentence pairs. Sentence embedding models other than Sent2Vec require pretrained word embeddings as the input. Over these input these models are trained and then generate the sentence vectors. In contrast, Sent2Vec performs both tasks simultaneously. It first generates the word embeddings and then combines them to generate the embedding for the sentence. Moreover, this is the state of the art model for unsupervised sentence embedding. Biosentvec gives the pretrained model for biomedical texts. However, the biosentvec model is not trained with all the words that appear in our collected data. That's why Sent2Vec is trained with various parameter settings over our data and the best model is chosen against a validation set which is a portion of the human annotated dataset from Houngbo and Mercer's work [26].

### 3.1.2 Annotated Sentence Pair Creation

As the citation linkage task is formulated as a sentence level textual semantic similarity measurement problem, a necessary step in creating the synthetic corpus is the gathering of pairs of citation and candidate cited sentences. To provide the cited sentences, among the 28,310 research documents, 112 were randomly selected from the biochemistry, cell biology and chemical biology domains. These papers were considered as the reference research papers while creating the corpus and conducting the downstream experiments. For these 112 research papers, 2289 papers which have cited them were manually collected. These papers were considered as the citing papers. From these papers only the corresponding citation sentences are extracted.

After cleaning the data, sentence pairs are generated, the first sentence in the pair being a sentence from the cited article and the second sentence being the citing sentence such that there is one sentence pair for each sentence in the cited article. This step generates 475,807 sentence pairs. Then, individual sentences of each sentence pair is fed to the already trained Sent2Vec model to get the vector representation. Then cosine similarity is measured between the sentence vectors for each sentence pair. The cosine similarity values come in the range from 0 to 1. For different cutoff cosine similarity values the performances are tested against the validation set. This validation set is a portion of Houngbo and Mercer's [26] human annotated corpus containing 800 sentence pairs (20 positive samples and 780 negative samples). To determine the best cutoff points, all the cutoffs are plotted on an ROC curve [45] and the best performance was found for cutoff value 0.57. Sentence pairs with the cosine similarity values

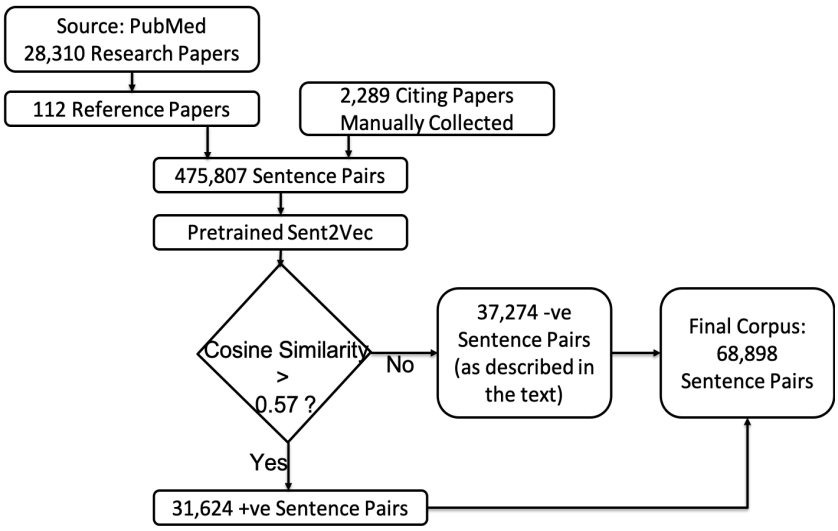



Figure 3.1: Annotated sentence pair creation for synthetic corpus build-up.

which are greater than or equal to the selected cutoff value are annotated with similarity value 1 and the remaining are annotated with 0. However, among these 475,807 sentence pairs most of the pairs are annotated with zero value. If the following models are trained with this data, they are highly likely to be biased. For this reason, among these sentence pairs 68,898 samples are chosen so that the dataset becomes balanced. Figure 3.1 shows the annotated sentence pair creation steps in a graphical way for better understanding. While choosing these data samples all of the sentence pairs annotated with similarity value 1 are kept. Then for each citation sentence, $n$ negative samples are chosen where $n$ is the number of positive samples found for that citation sentence. However, for some citation sentences, no positive sentence pair is found. For those citation sentences, five randomly chosen negative samples are inserted into the dataset. Thus, the dataset of 68,898 samples is created. This dataset contains 31,624 positively annotated sentence pairs which comprise 45.89% of the dataset.

Finally, for the validation set, 800 samples from the human annotated corpus are chosen. As the human annotated dataset is highly imbalanced and only 81 positive samples are present, we randomly chose 20 positive samples to use in the validation set. The test set is separated with 3057 samples which contain 61 positive samples. Each of the sets are non-overlapping.

### 3.1.3 Data Cleaning

The full-text articles that we have to conduct our experiments differ in format because of genre and journal writing styles. For instance, the same equation may appear in different contexts with different variable names and symbols. Since these names and symbols could possibly be used to make a semantic similarity decision it was decided to replace equations with a single

symbol. The same operation has been done for numbers as well, since the many different numbers would increase the vocabulary size and in many cases each number occurs infrequently. Knowing that a number is part of a sentence rather than knowing its exact value is usually what is needed for determining semantic similarity. Sentences may contain a few unnecessary symbols in terms of semantic representations that need to be deleted. Furthermore, the same symbol or operator is represented in different formats because of using different encoding systems while writing the documents. They need to be represented in a common format. Details of this data cleaning are given next.

Some sentences collected from different biomedical research articles are simply equations, nothing else. An analysis of the data suggested that identifying a few specific symbols (presented in Table 3.1) would be enough to recognize such equation sentences. These sentences are deleted. Citation numbers like "[xx]", which are unnecessary while computing the semantic relatedness at the sentence level, are also removed. Table 3.2 shows the regex commands used for this and other data cleaning steps that are discussed below.

All the equations that remain as part of a sentence are recognized and replaced with "<equ>". The regex for locating equations is constructed to consider various contexts. First, equations may start with various brackets like "(", "{", etc. Equations may contain mathematical operators like "∪", "∩", etc. Functions like "$f(g(x))$" need to be recognized. There may be special symbols for log functions or integration functions. There may be words or numbers as part of the equation. They may contain Greek letters as well. Following these operands there may or may not be some enclosing brackets. Following that, there may be zero, one or more parentheses. Following that, there can be any mathematical symbol. And finally, there may be some ending braces. So this regex can capture any equation with an optional first operand like "-" followed by pairs of mathematical symbols and operands.

There is occasional ambiguity with some of the characters. One case appears for the "*" sign. It may come as an operator in an equation or it may come out as a sign to represent some reference inside the text. The second case is not important for the semantic relatedness measurement task. For example, "*By contrast , SP reduced the topological uncertainty of*

Table 3.1: Symbols for deletion

| Operation | Symbol List |
|---|---|
| Delete the symbols | ™ ◒ ▯ ⑥ ▲ ♦ ≎ ④ ≙ , ◊ ◡ ◓ ⌐ ③ ! ∘ ② ¶ ο ] ˄ § ∀ ‖ ① ' ♠ [ ❘ ❘ ❖ \u2061 · ∆ ∘ ℗ ↓ ● ▶ ⟦ ℙ · ⊿ ˜ ¿ ‹ + · ♣ ∨ ∆ Ⅲ ❑ © ] ↑ ▫ ∏ ▪ ∫ ◯ ⟧ \ ‣ ■ \u200b ▼ \xad ¯ \u2062 ✓ ¨ † ♦ ¤ ➢ 〉 ® · ⑤ \u2061 ∃ ' |
| Delete the sentence | ⟩ ‿ ⟨ ⁀ □ ¯ ˙ ‖ ^ ⁖ |

Table 3.2: Regex commands for capturing different patterns throughout the data

| Purpose | Regex command |
|---|---|
| **Capture equations** | [⟨⟨{(\[\s]*([^\s⟨⟨{(\[⟩⟩)}\]%*+/:<=>\-\^\|~¬±·×÷ϴθ—<br>←→↔↦⇋⇔∀∇∈∉∈∌∏∑∓∘√∝∧∩∪∮≁≤≥≰⊂⊄⊆⊇⊈⊑⊕<br>⊙⊣⊤⊥∪·\|⇊▷○◦⟶⇊]+(\s*[⟨⟨{(\[]+\s*)+)?(∃\|∫\|log\|ln)?\s*<br>([^\s⟨⟨{(\[⟩⟩)}\]]+\|(\d+[\.,])+\d+)?[⟩⟩)}\]]*([⟨⟨{(\[\s]*[%*+/:<<br>=>\^\|~¬±·×÷ϴθ—<br>←→↔↦⇋⇔∀∇∈∉∈∌∏∑∓∘√∝∧∩∪∮≁≤≥≰⊂⊄⊆⊇⊈⊑⊕<br>⊙⊣⊤⊥∪·\|⇊▷○◦⟶⇊]+\s*-<br>?[⟨⟨{(\[\s]*(∃\|∫\|log\|ln)?\s*([^\s⟨⟨{(\[⟩⟩)}\]]+\|(\d+[\.,])+\d+)<br>\s*([⟨⟨{(\[\s]*[^\s⟨⟨{(\[⟩⟩)}\]%*+/:<=>\^\|~¬±·×÷ϴθ—<br>←→↔↦⇋⇔∀∇∈∉∈∌∏∑∓∘√∝∧∩∪∮≁≤≥≰⊂⊄⊆⊇⊈⊑⊕<br>⊙⊣⊤⊥∪·\|⇊▷○◦⟶⇊]+\|(\d+[\.,])+\d+\|-)?[\s⟩⟩)}\]]*)+ |
| **Capture numbers with no prior symbol** | (^\d+\| \K\d+)((\.\d+)\|\^\d+\|e[\-+]?\d+(\.\d+)?)?(?![\-\w]) |
| **Capture numbers with prior symbol** | (?<!(\w\|\d))[\-+]\d+((\.\d+)\|\^\d+\|e[\-+]?\d+(\.\d+)?)? |
| **Capture citations** | \[\d+([,\-]\d+)*\] |
| **Capture chemical names ending with ∝ and $\beta$** | -\s*∝\|-\s* $\beta$ |

*PAUP* to a large extent* .". In this case the "*" sign with the word "PAUP" is not something meaningful. In situations like this, this sign is deleted. Similarly, some ambiguities are found for the $\alpha$ sign. This symbol may appear in a equation to indicate in proportion to. Or it may appear with a chemical name like "AUCO-$\alpha$". In the case of chemical names, this sign is replaced with the term "alpha". The reason to do so is the presence of the chemical names where only this symbol is different and it indicates some other chemical name. But the symbol is kept as it is if it appears as a part of an equation.

Sentences may contain some numbers which are not a part of a equation. All of the standalone numbers are replaced with "<num>". Among them, some may be composed just of numerals and some may come with some symbols in front of them like "-5". To capture them, two different regexes are built. The reason behind making separate regexes is that some chemical names come with hyphens, numbers and other operators like "C-O2", "HER2". If the number appears as a part of any chemical name, they are preserved in their original form.

In the next step, all the symbols with different representation formats are replaced with

Table 3.3: Symbols and their corresponding replacement

| Symbol | Replacement | Symbol | Replacement | Symbol | Replacement | Symbol | Replacement |
|---|---|---|---|---|---|---|---|
| ³ | cubic | ‟ | “ | ″ | “ | ⇒ | = |
| ≋ | = | ‵ | ' | – | - | = | = |
| ≥ | >= | ∮ | <symbol> | ∝ | alpha | ↔ | = |
| € | Euro | × | <symbol> | ∮ | <symbol> | ″ | “ |
| $ | USD | ⋯ | - | ≡ | = | . | . |
| » | “ | ‥ | “ | → | <symbol> | 💥 | <symbol> |
| ‡ | <symbol> | 〉 | 〉 | ⋯ | <symbol> | ∧ | <symbol> |
| ¼ | one fourth | › | > | : | : | ² | square |
| ≃ | = | ¢ | cent | ‚ | , | ≧ | >= |
| ≳ | >= | ≦ | = | ? | ? | 〈 | 〈 |
| ▽ | <symbol> | * | * | « | “ | - | - |
| ‰ | percent | ) | ) | ” | “ | ≼ | <= |
| @ | @ | , | , | ∠ | angle of | ≔ | = |
| ≪ | << | – | - | ‚ | , | · | multiplied with |
| ° | Degree | ₦ | Nigerian Naira | ’ | ' | ¡ | <symbol> |
| ⌊ | lower limit of | ; | ; | ∅ | <symbol> | ⟨ | ⟨ |
| ∤ | !< | % | percent | ⅔ | two third of | ╫ | <symbol> |
| ≤ | <= | ≦ | <= | ≺ | < | # | # |
| — | - | ~ | = | ≠ | != | ⇜ | <symbol> |
| ≥ | >= | ¥ | Japanese Yen | + | plus | & | and |
| ‖ | “ | ℃ | Degree Celsius | ⅓ | one third of | ∞ | <symbol> |
| ⋯ | upto | ' | ' | < | < | - | - |
| 〉 | 〉 | { | { | ‘ | ' | 🄿 | |
| ▹ | <symbol> | ♂ | male | ~ | = | ½ | half |
| ≈ | = | C | C | ↡ | <symbol> | α | alpha |
| £ | Turkish Lira | ⋔ | <symbol> | ≲ | <= | μ | miu |
| “ | “ | } | } | ◇ | <symbol> | λ | lamda |
| ⊙ | <symbol> | - | - | ≫ | > | δ | sigma |
| £ | Pound | ∇ | <symbol> | ⩾ | >= | β | beta |
| ⊓ | <symbol> | ≅ | = | > | > | Ω | ohm |
| ♀ | female | ( | ( | ° | Degree | τ | tao |
| ɣ | gamma | Â | A | – | - | / | / |

their corresponding common format representation (Table 3.3). In the end, some unnecessary symbols are deleted. Finally all alphabetic characters are lower-cased.

## 3.2 Citation Linkage as a Semantic Similarity Measurement Task

Recalling from Section 3.1, the citation span is restricted to a single sentence. So, the citation linkage task is to find the set of sentences that are being referred to in the cited article by the citation sentence. With this in mind, the dataset discussed in Section 3.1 has been prepared

with pairs of sentences, the first sentence in the pair being a sentence from the cited article and the second sentence being the citation sentence. These sentence pairs have been automatically annotated in the manner discussed therein. The task for this dataset is to determine which of the sentence pairs are examples of citation linkage.

One measure of citation linkage is whether the citing and cited sentences are semantically similar. So, in this thesis, the citation linkage task is being viewed as a semantic relatedness measurement problem. Henceforth, the task at hand will be viewed simply as a semantic relatedness task. Furthermore, the semantic relatedness task can be viewed as a binary classification task: for each pair of sentences, are they semantically related or not.

In this thesis the semantic relatedness task is modelled using neural models. Section 3.2.2 discusses the supervised models that will be used for determining the semantic similarity of the sentence pairs. These models need vector representations of the words as input. The word embedding technique is discussed in Section 3.2.1.

### 3.2.1 Word Embedding for the Semantic Similarity Measurement Task

For the embedding of the words, fasttext [4] has been used. The reason for this choice is that this model has the ability to generate word vectors even if that word is unseen to it in training by utilizing the sub-word embeddings. Biowordvec, a word embedding specially developed for the biomedical domain, offers pretrained word embeddings for biomedical corpora. However, upon inspection, a lot of chemical names were absent in their pretrained embeddings. That's why we trained fasttext with our data. For the training of fasttext, all of the 4,843,746 sentences from the 28,310 documents used to train Sent2Vec together with the citation sentences collected from the 2289 papers used in the preparation of our citation linkage dataset are used. The fasttext model has been trained with various parameter settings (see Chapter 4 for details). The model with the best pearson and spearman values over the UMNSRS-Sim [49] dataset, a standard dataset used to evaluate biomedical word embeddings, is chosen for use in the semantic similarity task. This dataset contains 566 UMLS concept pairs manually rated for semantic similarity using a continuous response scale.

### 3.2.2 Neural Models for the Semantic Similarity Measurement Task

Of the various top-ranked neural models for the semantic similarity task, infersent has proven to be the best LSTM-based architecture on the standard semantic similarity tasks used to evaluate semantic similarity models. So, we have chosen the infersent [15] model to use in this thesis. For a better understanding of this architecture, this section first discusses the siamese architecture, a key component of infersent, and then the overall architecture.

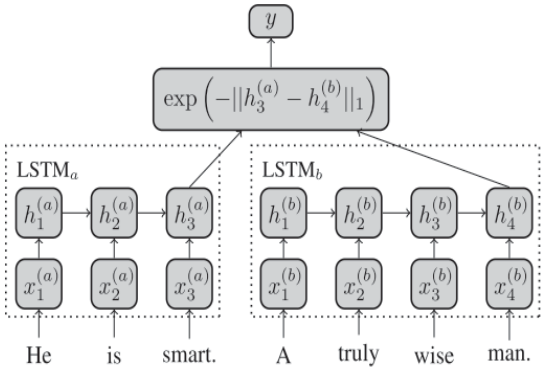



Figure 3.2: Siamese adaptation of the LSTM architecture for the semantic relatedness task (source: Mueller and Thyagarajan [46]).

In 2016, Mueller and Thyagarajan [46] adapted the siamese architecture [9] for the LSTM network for the semantic relatedness task. For this task they restricted the subsequent operations of the LSTM network to work with a simple Manhattan metric and this model outputs the vector representations of the sentences which can preserve complex semantic properties.

This model utilizes two LSTMs where each of them is responsible to process only one sentence of the given sentence pair. However, the trick they used is that both the LSTMs share the same architecture and weight matrices which makes this model more suitable for tasks in asymmetric domains. Figure 3.2 gives the overview of the model architecture.

The LSTM tries to map a variable length sequence of word vectors into a fixed length vector which eventually represents the sentence into a vector form. A sentence of $T$ words is passed to the LSTM. At each time step, the LSTM processes one word sequentially and updates its hidden state. At time step $T$, the hidden representation of the LSTM, $h^T \in \mathbb{R}^{d_sen}$, is the vector representation of that sentence. For a sentence pair, this model utilizes a similarity function $g$ over the LSTM representations. This similarity measurement metric determines the underlying semantic relatedness between the given sentence pair. The similarity function is defined as (Eqn. 3.1) [46]:

$$g(h_{T_a}^{(a)}, h_{T_b}^{(b)}) = \exp(-||h_{T_a}^{(a)} - h_{T_b}^{(b)}||) \in [0, 1] \tag{3.1}$$

where, $h_{T_a}^{(a)}$ and $h_{T_b}^{(b)}$ are the hidden state representations of sentences $S_a$ and $S_b$, respectively.

Later in 2017, Conneau et al. [15] investigated different neural network based techniques for generating fixed length vector representations of the sentences utilizing the siamese structure [46]. They trained their model over SNLI dataset with the concept that better semantic of a particular sentence can be preserved if the sentence encoding model is trained over natural language inference data. Their explored neural network based techniques are simple recurrent

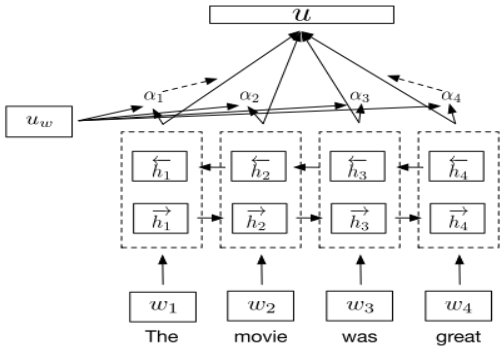



Figure 3.3: Infersent with Inner Attention (source: Conneau et al. [15]).

neural network (RNN), Long Short-Term Memory (LSTM), Bi-directional LSTM (Bi-LSTM), and gated recurrent unit (GRU) with different pooling operations, hierarchical CNN, inner-attention mechanism [36], hierarchical attention mechanism [63] and self-attentive network [35]. In their first attempt,they applied simple LSTM and GRU over a input sequence of $T$ words $(x_1, x_2, \ldots x_T)$ and at time step $T$, the sentence vector representation is generated in the form of hidden state representation. Experiments with Bi-LSTM and Bi-GRU output one hidden representation from the forward pass and one hidden state representation from the backward pass in both cases. They concatenated those two hidden vectors and pick either mean or max value from the vector for each time step. This operation outputs a vector of size $T$ that represents the sentence in the vector space. Later they tried with inner-attention mechanism over the hidden state representations of the Bi-LSTM unit (3.3) and generates the vector representation of the sentence ($u$). The procedure is as follows (Eqns. 3.2-3.4) [15]:

$$\bar{h}_i = \tanh(Wh_i + b_w) \tag{3.2}$$

$$\alpha_i = \frac{e^{\bar{h}_i^T u_w}}{\sum_i e^{\bar{h}_i^T u_w}} \tag{3.3}$$

$$u = \sum_t \alpha_i h_i \tag{3.4}$$

In their next attempt, they utilized the idea from the hierarchical attention mechanism over Bi-LSTM at sentence level. For this task, they applied the idea of the inner attention mechanism [36]. But, unlike the previous approach which computes the attention weighted sentence representation once, they computed it several times and finally concatenated them to generate the final sentence representation. They got their best result utilizing four context vectors.

They also used the structured self-attention mechanism over Bi-LSTM for the natural language inference task. Unlike the previous approaches, structured self-attention mechanism

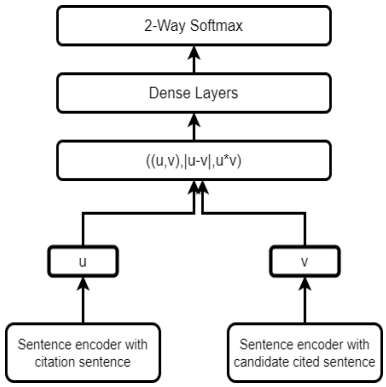



Figure 3.4: Infersent training mechanism

represents a sentence by a matrix instead of a vector. Because of using siamese architecture, two identical Bi-LSTMs with structured self-attention mechanism generate matrix formed sentence representations for both of the sentences in the pair. Then for each sentence, rows in the corresponding matrix are concatenated to form the sentence representation.

Being inspired by a hierarchical convolutional model named AdaSent [66], they introduced the Hierarchical ConvNet [15] which applies four consecutive convolution operations over the word vectors. At each layer max pooling is applied over the convoluted feature maps to generate one intermediate sentence representation. Finally, all four such intermediate representation vectors are concatenated to produce the final sentence representation. The architecture of the Hierarchical ConvNet is portrayed by Figure 3.5.

During the training of the models, two identical models are operated over two input sentences which output two vector representations for the sentences $(u, v)$. Then a feature map is generated which accommodates the concatenation, absolute point-wise difference and point-wise multiplication of these two sentence vectors. This feature map is then fed to the following dense and softmax layer to get the class prediction. After the model is trained, everything else in the sentence encoder part is discarded. Figure 3.4 shows the training process of the infersent model. After completing experiments with all of the models, the authors noted that among all the neural network architectures Bi-LSTM with max-pooling gives the best sentence encoding. An architectural overview of the infersent model with Bi-LSTM and max-pooling is illustrated in Figure 3.6.

### 3.2.3 Using Infersent for the Semantic Similarity Measurement Task

Recalling the discussion in the introduction to this section, the citation linkage task is being viewed as a semantic similarity measurement task where sentence pairs are annotated as being semantically similar or not. The dataset used for the experiments in Chapter 4 has been created

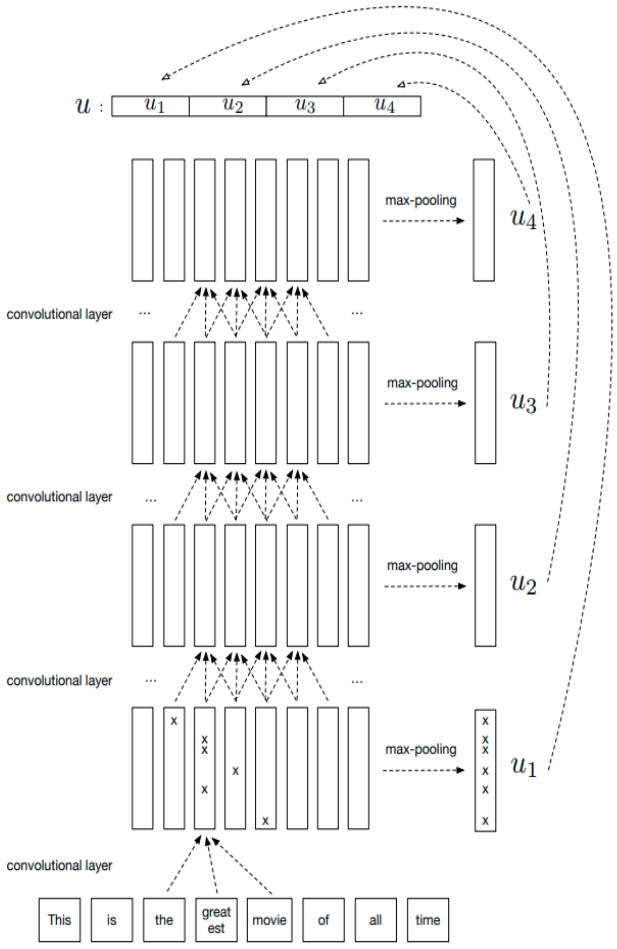



Figure 3.5: Hierarchical ConvNet architecture (source: Conneau et al. [15]).

with sentence pairs annotated with class label 1 to represent that the second sentence in the pair, the citing sentence, is referring to the first sentence in the pair, the cited sentence, and annotated with class label 0, if there is no reference. To decide whether the two sentences in the sentence pair are similar, the infersent architecture has been designed as a binary class classifier where class labels are 0 and 1.

For the semantic similarity measurement task five variants of the infersent architecture are used. The first one is the Bi-LSTM with max-pooling architecture as described earlier. In the second architecture, an inner attention mechanism [36] is applied over the outputs of the LSTM network. For the third attempt, an hierarchical attention mechanism [63] is applied over the output of the Bi-LSTM model. Hierarchical inner attention mechanism was introduced for document classification. For our purposes, only the first portion of their approach is applied. This approach makes it similar to the inner attention mechanism. The only difference in these two approaches is that just like their way of focusing on multiple important portions of the

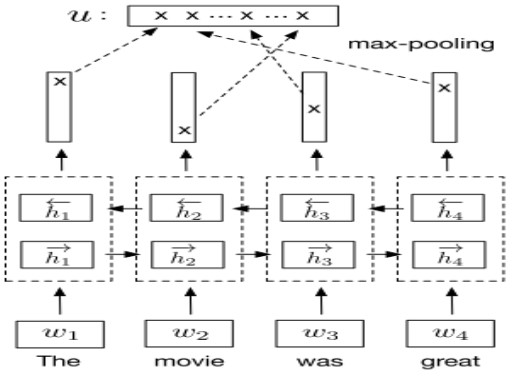



Figure 3.6: Infersent model with Bi-LSTM and max-pooling (source: Conneau et al. [15])

sentence [63], we used four context vectors instead of one like inner attention mechanism [36]. Thus this model generates four representations of the same sentence. These four different representations are then concatenated to generate the sentence representation. In the following attempt, Bi-LSTM with structured self-attention [35] is applied with four hops. For the last attempt,the hierarchical convolutional neural network (Hierarchical ConvNet) is used. For this task, four layers of the convolution operation with max pooling is applied. Max pooling is applied over the feature map generated at each convolution layer and a representation $u_i$ is generated. Concatenating this representation from each layer the sentence representation is generated. After that the same operations, as stated earlier for the final classification task, are applied for all the models. However, the best result is achieved for infersent with the hierarchical attention mechanism over Bi-LSTM architecture.

In the end, bootstrapping approach is utilized for the citation linkage task. The bootstrapping is performed with the variants of the Infersent architectures. For this task, the whole sentence pair dataset is separated into three portions. The annotation we got after running Sent2Vec is kept as it is for the first portion. Then with this data, the Infersent model is trained and validated against the human annotated validation data. The model with best validation accuracy is saved and used to annotate the second portion of the data. After annotating the second portion of the data, this data is added with the previously trained set and this combined data is used to train the Infersent model again. This time, the model is used to annotate the third portion of the data. After this third run is done, this annotated data is also added to the training set and Infersent is trained one more time with the latest trained dataset. The performance is then tested against the human annotated dataset. For a single bootstrapping, the same neural network architecture is used in all the trials, hence, five bootstrapping runs are made for the experiments.

# Chapter 4

# Experimental Setup And Results Analysis

This chapter concerns the details of the experimental setup for the semantic similarity measurement task and the analysis of the experimental results. It is divided into two parts. The first section discusses the parameter settings of the different word embeddings, sentence embeddings, and the five Infersent architectures used in the experiments. We used various parameter settings for the different models to obtain the optimal parameter configurations for these models. In the second section, the various models' performances with these optimal parameters are discussed in detail.

## 4.1 Network Parameters and Settings

For the dataset creation and semantic similarity measurement task discussed in Chapter 3, we trained one word embedding, one sentence embedding and four infersent architectures with different neural networks and attention mechanisms.

For the training of fasttext, we tried with both the skip-gram and CBOW architectures. We also varied the word embedding dimension, number of epochs, and window size. The n-grams at both the character and word levels were kept static to reduce the number of experiments. For both n-gram hyper-parameters, the value is set to 5 as in most of the experiments this n-gram value give the best result. Table 4.1 gives a list of different hyper-parameter configurations used for the training of the fasttext model. These different hyper-parameter settings were used for both skip-gram and CBOW architectures.

Finally, the fasttext model with the skip-gram architecture which generates 300 dimensional word vectors after 10 epochs having window size 5, maximum number of considered subwords 5, and learning rate 0.05 was selected, as the best pearson and spearman values over UMNSRS-Sim [49] are found with this parameter setting. The pearson and spearman values obtained for this parameter settings are 0.576 and 0.566. These values are comparably lower than BioWord-

Table 4.1: Hyper-parameter settings for the training of the fasttext model.

| **Hyper-parameter** | **Ranges** | **Selected** |
|---|---|---|
| **Embedding dimension** | 200/300/600 | 300 |
| **Epochs** | 5/7/10 | 10 |
| **Window size** | 5/10/20 | 5 |
| **Maximum number of subwords** | 5 | 5 |
| **Learning rate** | 0.01/0.05/0.1 | 0.05 |
| **Architecture type** | Skip-gram/CBOW | Skip-gram |

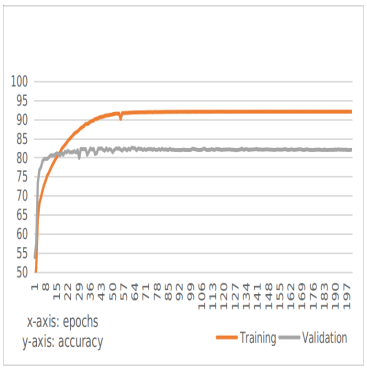
Table 4.2: The hyper-parameter setting for Sent2Vec sentence embedding architecture

| **Hyper-parameter** | **Ranges** | **Selected** |
|---|---|---|
| **Embedding dimension** | 200/300/400/500/600/700/800 | 500 |
| **Epochs** | 5/7/10/15/20 | 10 |
| **Window size** | 10/20 | 20 |
| **Learning rate (LR)** | 0.01/0.05/0.1/0.2 | 0.2 |
| **Number of Negative Samples** | 10 | 10 |
| **Loss function** | Negative sampling/ hierarchical softmax/ softmax | Negative sampling |
| **Sampling threshold** | 0.0001 | 0.0001 |

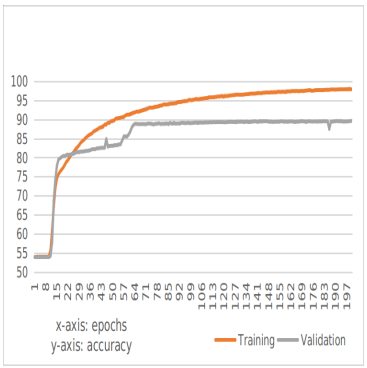
Vec which obtained 0.667 and 0.657, respectively. However, when final infersent models were run with BioWordVec the overall performance was not up to the mark. After analysing the data, we found that some chemical names were not present in the pre-trained BioWordVec vocabulary and due to our data cleaning process some chemical names are represented in ways that are different from theirs.

The Sent2Vec model incorporates the fasttext CBOW architecture in its own architecture and with these fasttext generated word vectors simultaneously generates the sentence vector. For the training of the Sent2Vec architecture, different hyper-parameter values were tested. Table 4.2 gives the list of different hyper-parameters. The best sentence embedding is found with a 500 dimensional vector representation.

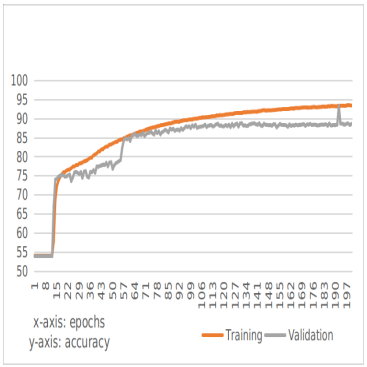
For the infersent architecture, the learning rate was set to 0.1. Gradient clipping was used while training. For a decrease in validation set accuracy, the learning rate accuracy was divided by 5. The batch size we tested with for all the architectures was 50 and the learning rate threshold was set to $10^{-5}$. For the final multi-layer perceptron, the hidden layer dimension was set to 512, and for the input of the LSTMs, 300 dimensional word embeddings were used. For the training of the architectures, stochastic gradient descent was used. For the infersent architecture these parameter settings are found in the original paper to give the best performance. To



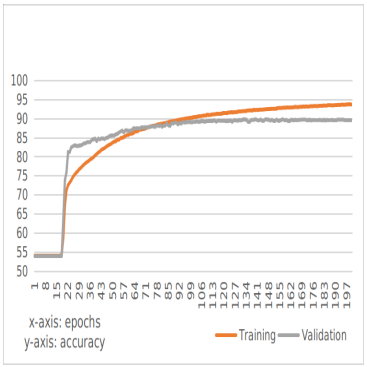

(a) Hierarchical ConvNet



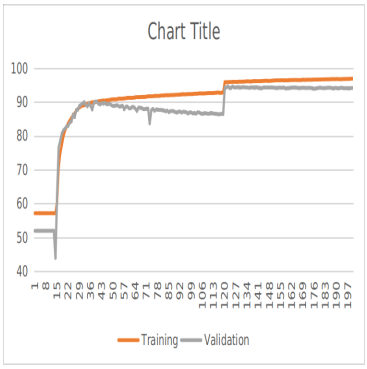

(b) Bi-LSTM with max-pooling




(c) Bi-LSTM with inner attention




(d) Bi-LSTM with hierarchical attention



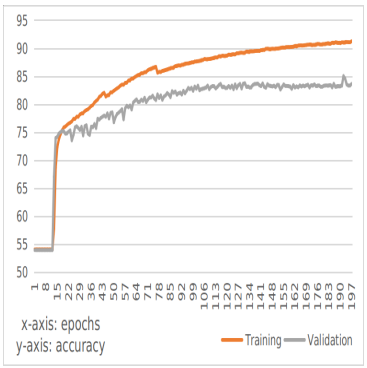

(e) Bi-LSTM with structured self attention

Figure 4.1: Training set and validation set accuracy for different models

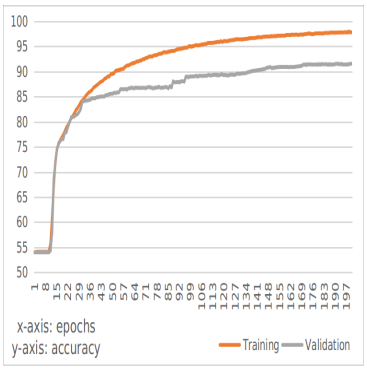
avoid a lot of experiments, these parameter values are used.

## 4.2 Performance Analysis

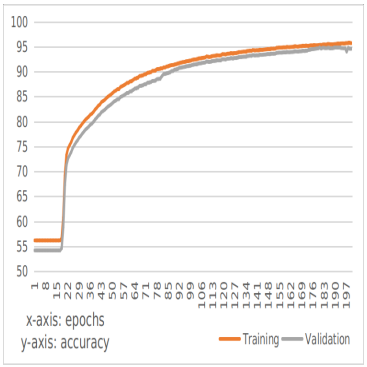
This section describes the results obtained from the experiments conducted for the semantic similarity measurement task. To analyze the performance of the word embedding models, pearson and spearman metrics are used. For the 300 dimensional word embedding, the calculated pearson and spearman values are 0.576 and 0.566 respectively.



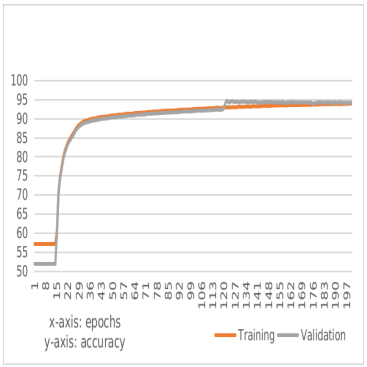

(a) Hierarchical ConvNet



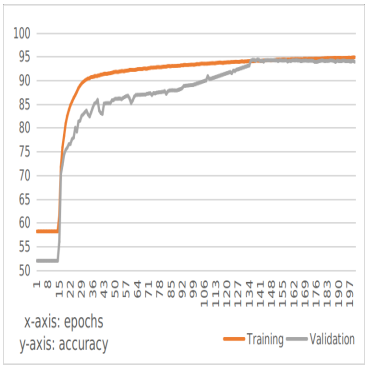

(b) Bi-LSTM with max-pooling




(c) Bi-LSTM with inner attention




(d) Bi-LSTM with hierarchical attention




(e) Bi-LSTM with structured self attention

Figure 4.2: Training set and validation set accuracy for different bootstrapped models

Figure 4.1 shows the training and validation accuracy of the individual models against epochs. From Figure 4.1(a) it is clear that the validation accuracy of the Hierarchical ConvNet model is a little bit lower compared to the validation set accuracy from the other models. The overall test accuracy of the model is also substantially lower compared to all of the other models (see Table 4.3). For the Bi-LSTM with max-pooling architecture, validation accuracy improves. Validation set accuracy improves in the same manner for the inner attention, structured self-attention and hierarchical attention mechanisms. For all other models apart from Hierarchical ConvNet, the validation accuracies are around 90%, whereas the Hierarchical ConvNet

is only 83.17%. Among the standalone models, the best validation accuracy is found for the Bi-LSTM with structured self-attention model. It is 94.22%. The worst performance is found for Hierarchical ConvNet. For Hierarchical ConvNet, the validation accuracy is 83.14%. Even Hierarchical ConvNet's training accuracy is less than 95%. Figure 4.2 shows the training and validation accuracy of these above stated models when they are bootstrapped. For the bootstrapped models the validation accuracy improves compared to their standalone forms. The best training accuracy is found for Bi-LSTM with max-pooling and that is 97.92%. However, the best validation accuracy is found for Bi-LSTM with hierarchical attention and Bi-LSTM with structured self-attention. In both cases, the validation accuracy is more than 95%. For Bi-LSTM with inner attention, the validation accuracy is 94.85%. Even in this scenario, the bootstrapped model with Hierarchical ConvNet comes with the lowest validation accuracy. The validation accuracy found for Hierarchical ConvNet is 85.27%. Still, the overall performance of the model improves if it is bootstrapped.

To analyze the performance of the models for the semantic similarity measurement task, precision, recall, accuracy, F1-score, false positive rate, true negative rate and false negative rate are the considered evaluation metrics. These metrics (Eqns. 4.1-4.7) [61] are defined by:

$$precision = \frac{tp}{tp + fp} \tag{4.1}$$

$$recall = \frac{tp}{tp + fn} \tag{4.2}$$

$$F1\text{-}score = \frac{2 * precision * recall}{precision + recall} \tag{4.3}$$

$$accuracy = \frac{tp + tn}{tp + fp + tn + fn} \tag{4.4}$$

$$true\ negative\ rate\ (TNR) = \frac{tn}{tn + fp} \tag{4.5}$$

$$false\ positive\ rate\ (FPR) = \frac{fp}{fp + tn} \tag{4.6}$$

$$false\ negative\ rate\ (FNR) = \frac{fn}{fn + tp} \tag{4.7}$$

However, for binary class classification with imbalanced datasets, the F-1 score doesn't reflect the performance of the model properly as it ignores true negative rate from consideration. Instead, metrics like the Matthews correlation coefficient (MCC) are more suitable when evaluating the performance of a binary classifier dealing with an imbalanced dataset [50]. The major advantage with MCC is that it considers true positives and negatives as well as false positives and negatives. Thus, it generates a balanced evaluation of the model's performance even if the dataset is highly imbalanced [7, 16, 40]. Given true and false positives and negatives it outputs

a value in the range of -1 to +1. Here, +1 indicates an accurate classification performance, -1 means there is no match between the predictions of the model and the true observations. If the MCC score is 0, then the model's performance is considered as nothing but random predictions. The MCC score for a binary classifier's classification performance over an imbalanced dataset is calculated using Eqn. 4.8 [61]:

$$MCC = \frac{tp \times tn - fp \times fn}{\sqrt{(tp + fp)(tp + fn)(tn + fp)(tn + fn)}} \tag{4.8}$$

Another way to assess the performance of a binary classifier when it works with an imbalanced dataset is balanced accuracy (BACC) [8]. The conventional accuracy metric can't properly evaluate the performance of a binary classifier if the dataset is highly imbalanced and the model is biased towards the class with most of the samples. In this scenario, if the model predicts the same class for all the samples in the dataset, the traditional accuracy metric would be equal to the proportion of the more frequent class's samples in the whole dataset, although the model has no capacity of generalization. As an example, if 90% of the samples of a dataset belong to class "A", the remaining 10% belong to class "B" and a binary classifier classifies all the samples as "A", the traditional accuracy of the classifier would be 90%. BACC overcomes this limitation of the traditional accuracy metric by considering the average recall gained by both of the classes. It outputs in the range (0,1) (it can also be represented as a percentage value). Higher BACC indicates better performance obtained by the model. The BACC is computed as (Eqn. 4.9) [8, 61]:

$$BACC = \frac{\frac{tp}{tp+fn} + \frac{tn}{tn+fp}}{2} \tag{4.9}$$

For the example given above, the BACC would be 0.45 (or 45%), which gives a better evaluation of the model's performance.

Table 4.3 shows the performance metrics found for different Infersent architectures and the bootstrapped models. We first trained and tested each model with two different settings. In the first setting, the models were provided with the pretrained BioWordVec word embeddings and the sentence pairs in the training set were annotated using the pretrained BioSentVec. In the second setting, the models were fed the fasttext word embeddings that we trained on our corpus. The sentence pairs in the training set were also annotated using Sent2Vec which was also trained on our corpus. Analysing the performance metrics from Table 4.3, it is clear that the models perform better in the second setup where both the word and sentence embedding models are trained on our own corpus. This happened for two reasons. First, the vocabulary of BioSentVec and BioWordVec doesn't contain all the words that appear in our corpus. Most of the time, the missing words are for the chemical names. The second reason is that BioSentVec

Table 4.3: Performance analysis of different models for the citation linkage task. The models are: M1: Hierarchical ConvNet, M2: Bi-LSTM with max-pooling, M3: Bi-LSTM with inner attention, M4: Bi-LSTM with hierarchical attention, M5: Bi-LSTM with structured self-attention. These five models have been trained with BioWordVec word embedding and BioSentVec sentence embedding. The notation used is "-Bio" after each method name. Furthermore, the five original methods are also bootstrapped. The notation used is "Boot-" placed in front of each method name. The column headings: TP and FP: true and false positives, respectively; TN and FN: true and false negatives, respectively; P: Precision; R: Recall; F1: F1-score; TNR: True Negative Rate, (True Positive Rate not shown because it is the same as Recall); FPR and FNR: False Positive and Negative Rate, respectively; MCC: Matthews correlation coefficient; Acc.: Accuracy; and BACC: Balanced accuracy.

| Model | TP | FP | TN | FN | P | R | F1 | FPR | TNR | FNR | MCC | Acc. (in %) | BACC (in %) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| M1 | 44 | 580 | 2416 | 17 | 0.07 | 0.72 | 0.13 | 0.19 | 0.81 | 0.27 | 0.18 | 80.5 | 76.38 |
| M2 | 53 | 365 | 2631 | 8 | 0.13 | 0.87 | 0.22 | 0.12 | 0.88 | 0.13 | 0.30 | 87.81 | 87.35 |
| M3 | 54 | 358 | 2638 | 7 | 0.13 | 0.89 | 0.22 | 0.12 | 0.88 | 0.11 | 0.31 | 88.07 | 88.28 |
| M4 | 55 | 356 | 2640 | 6 | 0.13 | 0.90 | 0.23 | 0.12 | 0.88 | 0.09 | 0.32 | 88.17 | 89.14 |
| M5 | 54 | 356 | 2640 | 7 | 0.13 | 0.89 | 0.23 | 0.12 | 0.88 | 0.11 | 0.31 | 88.15 | 88.32 |
| M1-Bio | 42 | 644 | 2352 | 19 | 0.06 | 0.69 | 0.11 | 0.21 | 0.79 | 0.31 | 0.16 | 78.32 | 73.68 |
| M2-Bio | 52 | 419 | 2577 | 9 | 0.11 | 0.85 | 0.20 | 0.14 | 0.86 | 0.15 | 0.28 | 86.01 | 85.63 |
| M3-Bio | 52 | 420 | 2576 | 9 | 0.11 | 0.85 | 0.20 | 0.14 | 0.86 | 0.15 | 0.28 | 85.99 | 85.61 |
| M4-Bio | 54 | 376 | 2620 | 7 | 0.13 | 0.89 | 0.22 | 0.13 | 0.87 | 0.11 | 0.31 | 87.48 | 87.98 |
| M5-Bio | 54 | 374 | 2622 | 7 | 0.13 | 0.89 | 0.22 | 0.12 | 0.88 | 0.11 | 0.31 | 87.56 | 88.02 |
| Boot-M1 | 46 | 576 | 2420 | 15 | 0.07 | 0.75 | 0.13 | 0.19 | 0.81 | 0.25 | 0.20 | 80.69 | 78.09 |
| Boot-M2 | 53 | 359 | 2637 | 8 | 0.13 | 0.87 | 0.22 | 0.12 | 0.88 | 0.13 | 0.31 | 88.02 | 87.45 |
| Boot-M3 | 54 | 349 | 2647 | 7 | 0.13 | 0.89 | 0.23 | 0.12 | 0.88 | 0.11 | 0.32 | 88.38 | 88.43 |
| Boot-M4 | 56 | 339 | 2657 | 5 | 0.14 | 0.92 | 0.25 | 0.11 | 0.89 | 0.08 | 0.34 | 88.75 | 90.24 |
| Boot-M5 | 56 | 343 | 2653 | 5 | 0.14 | 0.92 | 0.24 | 0.11 | 0.89 | 0.08 | 0.33 | 88.63 | 90.18 |
| Houngbo et al. [25] | 34 | 995 | 2001 | 27 | 0.03 | 0.56 | 0.06 | 0.33 | 0.66 | 0.44 | 0.07 | 66.58 | 61.26 |
| Li et al. [33] | 39 | 779 | 2217 | 22 | 0.05 | 0.64 | 0.09 | 0.26 | 0.74 | 0.36 | 0.12 | 73.81 | 68.97 |

was trained over sentences from the research article abstracts. These tend to be shorter in length compared to the sentences from the Method section of research articles which comprise our test set. Sentences in our training corpus are from different portions of the research articles and they tend to be longer in length. For all the models, the F-1 score, MCC, accuracy, and balanced accuracy are found to be better for the second setup. We then used these models, which use our trained fasttext and Sent2Vec, for the bootstrap models.

Among the five different Infersent architectures, the best result in terms of test set accuracy and F-1 score is found for Bi-LSTM with hierarchical attention mechanism and the worst performance is found for Hierarchical Convnet. For the Hierarchical Convnet, not only is its accuracy is lower, but also it captures fewer true positives. Out of 61 positive samples, it correctly captures only 44, whereas it captures 580 negatives samples as positive ones. It even captured

the most false negative samples (17). It captures 2416 negative samples properly. Both Bi-LSTM with the structured self-attention mechanism and the inner attention mechanism capture 54 positive samples correctly. However, the Bi-LSTM with self-attention mechanism comes with a higher accuracy as it captures more negative samples correctly. Among the bootstrapped architectures the best test set accuracy and F-1 score is found for Bi-LSTM with hierarchical attention. It captures 56 positive samples correctly with an F-1 score of 0.25. It also captures 2657 negative samples correctly. Both Infersent architectures with and without bootstrapping captures 56 positive samples correctly. However, the bootstrapped version classifies negative samples more accurately. The bootstrapped model with structured self-attention mechanism over Bi-LSTM also classifies 56 positive samples correctly. However, it comes with a little bit lower accuracy (88.63%). In terms of MCC and BACC, the best result is also found for the bootstrapped Bi-LSTM with hierarchical attention mechanism (0.34 and 90.24%, respectively). The bootstrapped Bi-LSTM with structured self-attention mechanism also gives higher MCC and BACC scores than the other models. The MCC and BACC scores found for this model are 0.33 and 90.18%, respectively. From the results with the bootstrapping approaches, it is clear that the performance has improved in all cases, though the improvements are not significant. In terms of MCC, the Bi-LSTM with hierarchical attention gives the best result among the simple models (0.32) as well as bootstrapped models (0.34).

To compare the performance of the model with the two pre-existing models, we have trained these two models with our synthetic corpus and then tested against the human annotated corpus. From Table 4.3 it is clearly visible that the models we have developed in this thesis surpass the previous works' performances. In their own work, Houngbo and Mercer [25] reported that their approach captured only 48.5% positive samples out of 81 whereas if their model is trained with our much larger synthetic dataset, it captures 55.73% positive samples which we can say is a good improvement. This result gives proof of the fact that our contributing this synthetic dataset has been important for the citation linkage task research in the biomedical domain. We report the retrained Li et al. model [33] but do not compare with the original work because their task was performed on computational linguistics research articles.

# Chapter 5

# Conclusions and Future Work

Citations form a network of connected research articles. In the case of scientific research articles, a citation refers to the document from where the idea stated in the citing sentence originates. For a number of purposes that have been discussed in Section 1, being able to determine a more precise focus of a citation would be advantageous. This thesis has provided a neural learning method for finding those sentences in a cited article that are the focus of a citation in a citing paper. This final chapter summarizes the contributions of this thesis, points to the short comings of the study, and gives directions for future work.

## 5.1 Conclusions

This thesis looks at one method to establish a relationship between the citation sentence and its corresponding reference sentences from the cited paper, a task called *citation linkage*. In this study, citation linkage is modelled as a textual semantic relatedness measurement task. The text span chosen for this semantic relatedness measurement task is the sentence. The task is formulated as a textual matching operation between a citation sentence and every sentence in the corresponding cited paper. A deep learning model is trained and evaluated as the method for this task.

The main contributions of this thesis are:

1. building a framework to determine the appropriate cited sentences from a cited paper given a citation sentence,

2. building a corpus for citation linkage task containing more than sixty thousand sentence pairs from the biomedical domain, and

3. developing a method for cleaning and preprocessing sentences from different biomedical domains

With respect to the first contribution, we have provided a technique to determine the appropriate cited sentences from a cited paper given a citation sentence that has significantly outperformed previous methods [26] designed for the biomedical domain and evaluated on that test data set and previous methods [34] that were designed for a different domain. The best infersent model with bootstrapping has achieved an accuracy of 88.75 on the biochemistry sentence pair data set provided by Hospice and Mercer [26] compared with 66.58 that was achieved by our implementation of the Hospice and Mercer model [25, 26] and trained on the synthetic training set and 73.81 that was achieved by [34] on a computational linguistics data set.

Deep learning models can learn the inherent pattern from the data throughout the training process. However, they require a lot of data for the training. As for this task no such large dataset is available, for our second contribution, we built a synthetic dataset for the training of our models. In the end, the quality of our synthetic dataset and the used models is tested against the human annotated dataset and our models surpass the previous model in terms of determining the relevant cited sentences from the referenced paper. As noted above, the use of this data set to train the models provided by Houngbo and Mercer [25, 26] improved their results as well. This is good evidence of the quality of this synthetic data set.

Regarding the third contribution, in order to provide sufficient data to train Sent2Vec, which was used to annotate our synthetic data set, we needed to clean and preprocess biomedical full-text articles. A number of regexes have been developed and are made available to the community.

The synthetic corpus, data cleaning tool, and citation linkage tool are available on GitHub (https://github.com/sudipta90/CitationLinkage).

## 5.2 Future Work

This thesis has suggested a method to make a link between citation sentences and sentences in the cited article based on semantic similarity of sentence pairs. This final section suggests directions to remove some of the assumptions made in this study and to improve the accuracy of the method proposed here.

A citation sentence refers to text spans in the referred text document. This text span may be either a part of a sentence, a single sentence, a paragraph, or a section. For our experiments, we worked at the sentence level only. This is a limitation of our work. Moving to a sub-sentence

level would need to use knowledge about sentence structure. We used sentence embeddings as the representation in the semantic similarity measurement task. Using techniques to embed larger text units would be an obvious generalization of this idea.

In addition to modifying the referred to text, one could consider more than the citation sentence itself in the citation linkage task. While annotating the human annotated test dataset, the annotator had some context knowledge. She annotated the data having some ideas about what has been written previous to the citation sentence and she also had biochemisty knowledge. This extra information can provide reasons why our model couldn't capture more than 74 positively annotated sentences among the 81. Moving beyond simple sentence similarity as a proxy for citation linkage to include this extra information is something to consider, but is something that is will require just NLP techniques.

The test data set that was used in this study was created only for method citation sentences. We have obtained good results having trained on a broader set of citations. It would be appropriate to human annotate a test set with a variety of citation types and see how good the proposed method performs on this expanded test set. Additionally, future research could also look at a wider subset of biomedical subdomains rather than the ones used in this study.

Recalling that the experimental biomedical article is written in the IMRaD style, Hospice and Mercer [25, 26] used this knowledge to improve the accuracy of their model. The citations in the test set are sentences that mention methods. So by reducing the set of possible citation linkage candidates to those that can be rhetorically categorized as method sentences, they were able to improve the performance of their methods. This would be something to look at when using the technique used in this thesis. (We attempted to obtain the reduced data set without success. So, this idea would require recreating that data set.) Given the suggested future work in the previous paragraph, it would be interesting to see how having the IMRaD rhetorical information would affect the performance.

This framework has the ability to work at the sentence level semantic relatedness measurement task. However, in future this semantic similarity measurement task can be performed at a larger text span level like paragraph. Furthermore, the models used here don't consider any kind of parsing. As a result chemical names with multiple words are being considered as separate entity. For an example, "sodium chloride" is being considered as two entities: "sodium" and "chloride". Using tree structured models like tree LSTM [1] might improve the performance.

# Bibliography


[1] Mahtab Ahmed, Muhammad Rifayat Samee, and Robert E Mercer. Improving tree-LSTM with tree attention. In *Proceedings of the 2019 IEEE 13th International Conference on Semantic Computing (ICSC)*, pages 247–254, 2019.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[3] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2003.

[4] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[5] Serena Bonin, F Petrera, B Niccolini, and Giorgio Stanta. PCR analysis in archival postmortem tissues. *Molecular Pathology*, 56(3):184–186, 2003.

[6] Lutz Bornmann, K Brad Wray, and Robin Haunschild. Citation concept analysis (CCA): A new form of citation analysis revealing the usefulness of concepts for other researchers illustrated by two exemplary case studies including classic books by Thomas S. Kuhn and Karl R. Popper. *Scientometrics*, 122:1051–1074, 2020.

[7] Sabri Boughorbel, Fethi Jarray, and Mohammed El-Anbari. Optimal classifier for imbalanced data using Matthews correlation coefficient metric. *PloS ONE*, 12(6), 2017.

[8] Kay Henning Brodersen, Cheng Soon Ong, Klaas Enno Stephan, and Joachim M Buhmann. The balanced accuracy and its posterior distribution. In *Proceedings of the 2010 20th International Conference on Pattern Recognition*, pages 3121–3124, 2010.

[9] Jane Bromley, James W Bentz, Léon Bottou, Isabelle Guyon, Yann LeCun, Cliff Moore, Eduard Säckinger, and Roopak Shah. Signature verification using a "siamese" time delay

neural network. *International ]ournal of Pattern Recognition and Artificial Intelligence*, 7(4):669–688, 1993.

[10] Jason Brownlee. Encoder-decoder recurrent neural network models for neural machine translation. https://machinelearningmastery.com/encoder-decoder-recurrent-neural-network-models-neural-machine-translation/. Accessed: 2019-12-11.

[11] Jason Brownlee. A gentle introduction to exploding gradients in neural networks. https://machinelearningmastery.com/exploding-gradients-in-neural-networks/. Accessed: 2019-12-11.

[12] Jason Brownlee. How to develop a bidirectional LSTM for sequence classification in python with keras. https://machinelearningmastery.com/develop-bidirectional-lstm-sequence-classification-python-keras/. Accessed: 2019-12-11.

[13] Antoni Buades, Bartomeu Coll, and Jean-Michel Morel. A non-local algorithm for image denoising. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 60–65, 2005.

[14] Qingyu Chen, Yifan Peng, and Zhiyong Lu. BioSentVec: Creating sentence embeddings for biomedical texts. In *Proceedings of the 2019 IEEE International Conference on Healthcare Informatics (ICHI)*, pages 1–5, 2019.

[15] Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 670–680, 2017.

[16] Harald Cramér. *Mathematical Methods of Statistics*, volume 9 of *Mathematical Methods of Statistics (PMS-9)*. Princeton University Press, 1946.

[17] easy tensorflow. Vanilla RNN for digit classification. http://www.easy-tensorflow.com/tf-tutorials/recurrent-neural-networks/vanilla-rnn-for-classification. Accessed: 2019-12-11.

[18] Eugene Garfield. "Science Citation Index"—A new dimension in indexing. *Science*, 144(3619):649–654, 1964.

[19] Eugene Garfield. Citation analysis as a tool in journal evaluation. *Science*, 178(4060):471–479, 1972.

[20] Mark Garzone and Robert E Mercer. Towards an automated citation classifier. In *Proceedings of the 13th Biennial Conference of the Canadian Society for Computational Studies of Intelligence*, pages 337–346, 2000.

[21] Hua He, Kevin Gimpel, and Jimmy Lin. Multi-perspective sentence similarity modeling with convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1576–1586, 2015.

[22] Hua He, John Wieting, Kevin Gimpel, Jinfeng Rao, and Jimmy Lin. UMD-TTIC-UW at SemEval-2016 Task 1: Attention-based multi-perspective convolutional neural networks for textual similarity measurement. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1103–1108, 2016.

[23] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, and Jürgen Schmidhuber. Gradient flow in recurrent nets: The difficulty of learning long-term dependencies. In *A Field Guide to Dynamical Recurrent Networks*, pages 237–243. IEEE Press, 2001.

[24] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[25] Hospice Houngbo and Robert E Mercer. Investigating citation linkage with machine learning. In *Proceedings of the 30th Canadian Conference on Artificial Intelligence*, pages 78–83, 2017.

[26] Kokou Hospice Houngbo. *Investigating Citation Linkage Between Research Articles*. PhD thesis, The University of Western Ontario, 2017.

[27] Sven E Hug, Michael Ochsner, and Martin P Brändle. Citation analysis with Microsoft Academic. *Scientometrics*, 111(1):371–378, 2017.

[28] Cornelis JJ Huijsmans, Jan Damen, Johannes C van der Linden, Paul HM Savelkoul, and Mirjam HA Hermans. Comparative analysis of four methods to extract DNA from paraffin-embedded tissues: Effect on downstream molecular applications. *BMC Research Notes*, 3:239, 2010.

[29] i2tutorials. Deep dive into bidirectional LSTM. https://www.i2tutorials.com/technology/deep-dive-into-bidirectional-lstm/. Accessed: 2019-12-11.

[30] K Kayser, H Stute, J Lübcke, and U Wazinski. Rapid microwave fixation—a comparative morphometric study. *The Histochemical Journal*, 20(6-7):347–352, 1988.

[31] Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Skip-thought vectors. In *Advances in Neural Information Processing Systems*, pages 3294–3302, 2015.

[32] Ceshine Lee. Understanding Bidirectional RNN in PyTorch. https://towardsdatascience.com/understanding-bidirectional-rnn-in-pytorch-5bd25a5dd66. Accessed: 2019-12-11.

[33] Lei Li, Liyuan Mao, Yazhao Zhang, Junqi Chi, Taiwen Huang, Xiaoyue Cong, and Heng Peng. Computational linguistics literature and citations oriented citation linkage, classification and summarization. *International Journal on Digital Libraries*, 19(2–3):173–190, 2018.

[34] Lei Li, Yingqi Zhu, Yang Xie, Zuying Huang, Wei Liu, Xingyuan Li, and Yinan Liu. CIST@CLSciSumm-19: Automatic scientific paper summarization with citances and facets. In *Proceedings of the 4th Joint Workshop on Bibliometric-enhanced Information Retrieval and Natural Language Processing for Digital Libraries (BIRNDL 2019)*, volume 2414 of *CEUR Workshop Proceedings*, 2019.

[35] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*, 2017.

[36] Yang Liu, Chengjie Sun, Lei Lin, and Xiaolong Wang. Learning natural language inference using bidirectional LSTM model and inner-attention. *arXiv preprint arXiv:1605.09090*, 2016.

[37] Lajanugen Logeswaran and Honglak Lee. An efficient framework for learning sentence representations. In *6th International Conference on Learning Representations (ICLR)*, 2018.

[38] Michael H MacRoberts and Barbara R MacRoberts. The mismeasure of science: Citation analysis. *Journal of the Association for Information Science and Technology*, 69(3):474–482, 2018.

[39] Tyler Martindale. How do I get started? Using citation analysis to become an effective liaison. Poster for Association of College & Research Libraries (ACRL), 2019.

[40] Brian W Matthews. Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta (BBA) - Protein Structure*, 405(2):442–451, 1975.

[41] Robert Mercer. Locating and extracting key components of argumentation from scholarly scientific writing. In Elena Cabrio, Graeme Hirst, Serena Villata, and Adam Wyner, editors, *Natural Language Argumentation: Mining, Processing, and Reasoning over Textual Arguments (Dagstuhl Seminar 16161)*, volume 6(4), chapter 3.15, pages 93–94. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016.

[42] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[43] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, 2013.

[44] Tsendsuren Munkhdalai, John Lalor, and Hong Yu. Citation analysis with neural attention models. In *Proceedings of the Seventh International Workshop on Health Text Mining and Information Analysis*, pages 69–77, 2016.

[45] Sarang Narkhede. Understanding AUC - ROC curve. https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5. Accessed: 2019-12-11.

[46] Paul Neculoiu, Maarten Versteegh, and Mihai Rotaru. Learning text similarity with siamese recurrent networks. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 148–157, 2016.

[47] Christopher Olah. Understanding LSTM networks. https://colah.github.io/posts/2015-08-Understanding-LSTMs/. Accessed: 2019-12-11.

[48] Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. Unsupervised learning of sentence embeddings using compositional n-gram features. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 528–540, 2018.

[49] Serguei Pakhomov, Bridget McInnes, Terrence Adam, Ying Liu, Ted Pedersen, and Genevieve B Melton. Semantic similarity and relatedness between clinical terms: An experimental study. In *Proceedings of the American Medical Informatics Association (AMIA) Annual Symposium*, pages 572–576, 2010.

[50] Pierre Perruchet and Ronald Peereman. The exploitation of distributional information in syllable processing. *Journal of Neurolinguistics*, 17(2-3):97–119, 2004.

[51] Dragomir R. Radev, Hongyan Jing, and Malgorzata Budzikowska. Centroid-based summarization of multiple documents: Sentence extraction, utility-based evaluation, and user studies. In *Proceedings of the NAACL-ANLP 2000 Workshop: Automatic Summarization*, pages 21–30, 2000.

[52] Hojjat Salehinejad, Sharan Sankar, Joseph Barfett, Errol Colak, and Shahrokh Valaee. Recent advances in recurrent neural networks. *arXiv preprint arXiv:1801.01078*, 2017.

[53] Niranjan J Sathianathen, Robert Lane 3rd, Benjamin Condon, Declan G Murphy, Nathan Lawrentschuk, Christopher J Weight, and Alastair D Lamb. Early online attention can predict citation counts for urological publications: The #UroSoMe_Score. *European Urology Focus*, Available online 6 November 2019.

[54] Mike Schuster and Kuldip Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45:2673–2681, 12 1997.

[55] D Shashavali, V Vishwjeet, Rahul Kumar, Gaurav Mathur, Nikhil Nihal, Siddhartha Mukherjee, and Suresh Venkanagouda Patil. Sentence similarity techniques for short vs variable length text using word embeddings. *Computación y Sistemas*, 23(3):999–1004, 2019.

[56] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. A latent semantic model with convolutional-pooling structure for information retrieval. In *Proceedings of the 23rd ACM International Conference on Information and Knowledge Management*, pages 101–110, 2014.

[57] Noah A Smith and Jason Eisner. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 354–362, 2005.

[58] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Advances in Neural Information Processing Systems*, pages 3104–3112, 2014.

[59] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[60] Yuker Wang, Victoria EH Carlton, George Karlin-Neumann, Ronald Sapolsky, Li Zhang, Martin Moorhead, Zhigang C Wang, Andrea L Richardson, Robert Warren, Axel Walther, Melissa Bondy, Aysegul Sahin, Ralf Krahe, Musaffe Tuna, Patricia A Thompson, Paul T Spellman, Joe W Gray, Gordon B Mills, and Malek Faham. High quality copy number and genotype data from FFPE samples using molecular inversion probe (MIP) microarrays. *BMC Medical Genomics*, 2(1):8, 2009.

[61] Wikipedia. Evaluation of binary classifiers. https://en.wikipedia.org/wiki/Evaluation_of_binary_classifiers. Accessed: 2019-12-11.

[62] Petra Wolffs, Halfdan Grage, Oskar Hagberg, and Peter Rådström. Impact of DNA polymerases and their buffer systems on quantitative real-time PCR. *Journal of Clinical Microbiology*, 42(1):408–411, 2004.

[63] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, 2016.

[64] Lin Yao, Zhengyu Pan, and Huansheng Ning. Unlabeled short text similarity with LSTM encoder. *IEEE Access*, 7:3430–3437, 2019.

[65] Yijia Zhang, Qingyu Chen, Zhihao Yang, Hongfei Lin, and Zhiyong Lu. BioWordVec, improving biomedical word embeddings with subword information and MeSH. *Scientific Data*, 6(1):52, 2019.

[66] Han Zhao, Zhengdong Lu, and Pascal Poupart. Self-adaptive hierarchical sentence model. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*, pages 4069–4076, 2015.

# Curriculum Vitae

| | |
|---|---|
| **Name:** | Sudipta Singha Roy |
| **Post-Secondary Education and Degrees:** | B.Sc. in Computer Science and Engineering<br>Khulna University of Engineering & Technology<br>2010 - 2014 |
| **Honours and Awards:** | Western Graduate Research Scholarship<br>2018-2019 |
| **Related Work Experience:** | Teaching Assistant<br>The University of Western Ontario<br>2018 - 2019<br><br>Research Assistant<br>The University of Western Ontario<br>2018 - 2019 |

**Publications:**

- Sudipta Singha Roy, Robert E. Mercer, and Felipe Urra, Investigating Citation Linkage as a Sentence Similarity Measurement Task Using Deep Learning, 33rd Canadian Conference on Artificial Intelligence, 2020. (In Press)

- Sudipta Singha Roy, Mahtab Ahmed, M. A. H. Akhand, Noisy Image Classification Using Hybrid Deep Learning Methods, Journal of Information and Communication Technology, 17(2):233–269, 2018.

- Sudipta Singha Roy, Sk Imran Hossain, MAH Akhand, Kazuyuki Murase, A robust system for noisy image classification combining denoising autoencoder and convolutional neural network, International Journal of Advanced Computer Science and Applications, 9(1):224–235, 2018.

- Shaikh Akib Shahriyar, Kazi Md Rokibul Alam, Sudipta Singha Roy, Yasuhiko Morimoto, An Approach for Multi Label Image Classification Using Single Label Convolutional Neural Network, 21st International Conference of Computer and Information Technology (ICCIT), pp. 1–6, 2018.

- Sudipta Singha Roy, Shaikh Akib Shahriyar, Md Asaf-Uddowla, Kazi Md Rokibul Alam, Yasuhiko Morimoto, A novel encryption model for text messages using delayed chaotic neural network and DNA cryptography, 20th International Conference of Computer and Information Technology (ICCIT), pp. 1–6, 2017.

- Sudipta Singha Roy, Mahtab Ahmed, MAH Akhand, Classification of massive noisy image using auto-encoders and convolutional neural network, 8th International Conference on Information Technology (ICIT), pp. 971–979, 2017.

- Sudipta Singha Roy, Tamjid Haque Sarker, MMA Hashem, A novel trust measurement system for cloud-based marketplace, 2nd International Conference on Electrical Information and Communication Technologies (EICT), pp. 49–54, 2015.