EXTRACTING PROTEIN-LIGAND INTERACTIONS FROM THE BIOMEDICAL LITERATURE USING DEEP LEARNING APPROACHES

by

Atakan Yüksel

B.S., Computer Engineering, Bilkent University, 2015

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Computer Engineering

Boğaziçi University

2019

EXTRACTING PROTEIN-LIGAND INTERACTIONS FROM THE BIOMEDICAL LITERATURE USING DEEP LEARNING APPROACHES

APPROVED BY:

Assoc. Prof. Arzucan Özgür .....
(Thesis Supervisor)

Assoc. Prof. Elif Özkırımlı Ölmez .....
(Thesis Co-supervisor)

Assist. Prof. Fatma Başak Aydemir .....

Assist. Prof. Öznur Taştan Okan .....

DATE OF APPROVAL: 11/07/2019

ACKNOWLEDGEMENTS

I would like to thank my mother Çiğdem Yüksel, my father Hikmet Yüksel and my girlfriend Büşra Koç for supporting me mentally when I'm feeling down.

I want to express my gratitude to my supervisor Arzucan Özgür and co-supervisor Elif Özkırımlı Ölmez for always helping me to pursuit my goals. It was my privilege to be their mentee.

ABSTRACT

EXTRACTING PROTEIN-LIGAND INTERACTIONS FROM THE BIOMEDICAL LITERATURE USING DEEP LEARNING APPROACHES

Protein-ligand interactions play crucial roles in living organisms, thus they attract many researchers from various disciplines. There are protein-ligand interaction databases that provide information to researchers in a suitable format. These databases extract the interactions manually from biomedical literature but the extraction process is becoming harder each day because of the increase in the number of biomedical publication, thereby the need for an automated extraction system has arisen. The aim of this thesis is to fulfill this need via deep learning models. This thesis includes performance analysis of Convolutional Neural Network (CNN) and Bidirectional Long Short Term Memory (BiLSTM) Networks for the task of protein-ligand interaction extraction. Comparison of features in terms of their effect on the performance of the models is also included in the thesis. The gold standard corpus that is created for BioCreative VI ChemProt task is selected as our dataset for training and evaluation of our models. Word embeddings, distance embeddings, part of speech (POS) tags and inside outside beginning (IOB) chunk tags are used as features in the models. The grid search algorithm is applied to find the optimal hyperparameters for each model in the experiments. The best models and input representations are selected via using the development set then they are evaluated on the test set. Based on the results on the test set, we concluded that BiLSTM performs better than CNN for each evaluated feature setting.

ÖZET

DERİN ÖĞRENME YAKLAŞIMLARI İLE BİYOMEDİKAL LİTERATÜRÜNDEN PROTEİN-LİGAND ETKİLEŞİMLERİNİ ÖĞRENME

Protein-ligand etkileşimi canlı organizmalarda çok önemli bir rol oynar, bu nedenle çeşitli disiplinlerden birçok araştırmacının ilgisini çeker. Araştırmacılara istenilen formatta bilgi sağlayan protein-ligand etkileşim veritabanları vardır. Bu veritabanları manuel olarak biyomedikal literatüründen çıkarılmaktadır ancak biyomedikal alandaki yayınların sayısındaki artıştan dolayı bu işlem her geçen gün daha da zorlaşmaktadır, dolayısıyla otomatik olarak protein-ligand etkileşimlerini metinlerden çıkaran bir sisteme ihtiyaç duyulmuştur. Bu tezin amacı bu ihtiyacı derin öğrenme modelleri ile karşılamaktır. Tez, Evrişimli Sinir Ağı (CNN) ve Uzun/Kısa Süreli Belleğin (LSTM) protein-ligand etkileşimi için performans analizini içerir. Ayrıca, farklı verilerin modellerin performansına etkisi bakımından karşılaştırılması da tez kapsamındadır. BioCreative VI ChemProt yarışması için oluşturulan derlemi, modellerimizin eğitimi ve değerlendirilmesi için veri seti olarak seçildi. Kelime temsilleri, mesafe temsilleri, cümle öğelerinin temsilleri ve iç dış başlangıç öbek temsilleri modelde özellik olarak kullanılmaktadır. Grid arama algoritması, deneylerde her model için optimal hiperparametreleri bulmak için uygulanır. En iyi modeller ve girdi gösterimleri geliştirme seti kullanılarak seçilir ve sonra test seti ile değerlendirilir. Test setindeki sonuçlara dayanarak BiLSTM'in her durumda CNN'den daha iyi performans gösterdiği sonucuna vardık.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS . . . iii
ABSTRACT . . . iv
ÖZET . . . v
LIST OF FIGURES . . . viii
LIST OF TABLES . . . x
LIST OF SYMBOLS . . . xiii
LIST OF ACRONYMS/ABBREVIATIONS . . . xiv
1. INTRODUCTION . . . 1
2. BACKGROUND . . . 3
2.1. Protein Ligand Interaction . . . 3
2.2. Protein Ligand Interaction Databases . . . 6
2.2.1. BindingDB . . . 6
2.2.2. PDBbind . . . 8
2.2.3. BindingMOAD . . . 9
2.3. Deep Learning . . . 9
2.3.1. Long Short Term Memory . . . 11
2.3.2. Convolutional Neural Network . . . 13
2.3.3. Word2Vec . . . 14
2.4. Related Work . . . 18
3. MATERIALS AND METHODS . . . 20
3.1. Dataset . . . 20
3.2. Input Representation . . . 21
3.3. Machine Learning Models . . . 26
3.3.1. Bidirectional LSTM . . . 26
3.3.2. Convolutional Neural Network . . . 29
3.4. Implementation . . . 30
3.4.1. Data Processing Component . . . 32
3.4.1.1. Dataset Class . . . 32

3.4.1.2. Data Interface Class 33
3.4.2. Deep Learning Component 33
3.4.2.1. BiLSTM model class 33
3.4.2.2. CNN model class 34
3.4.3. Predictor 34
3.4.4. Drivers 35
3.4.4.1. 5x2 Cross Validation Paired t test 35
3.4.4.2. Grid Search 35
4. EXPERIMENTS AND RESULTS 36
4.1. Training Domain of Word Embeddings 36
4.1.1. Bidirection LSTM 37
4.1.2. Convolutional Neural Network 39
4.2. Trainable Word Embeddings 40
4.2.1. Bidirectional LSTM 41
4.2.2. Convolutional Neural Network 42
4.3. Position Embedding Experiments 43
4.3.1. Bidirectional LSTM 44
4.3.2. Convolutional Neural Network 45
4.4. POS Tag Embedding Experiments 47
4.4.1. Bidirectional LSTM 48
4.4.2. Convolutional Neural Network 49
4.5. IOB Chunk Embedding Experiments 51
4.5.1. Bidirectional LSTM 51
4.5.2. Convolutional Neural Network 53
4.6. Test Set Evaluations 55
5. CONCLUSION 58
REFERENCES 61

LIST OF FIGURES

Figure 1.1. Number of indexed articles versus Fiscal year 1

Figure 2.1. Core structure of an amino acid. 3

Figure 2.2. Eclipsed and staggered conformations of ethane. 5

Figure 2.3. Example perceptron unit. 10

Figure 2.4. LSTM cell with two gates. 11

Figure 2.5. LSTM cell with three gates. 12

Figure 2.6. Simple convolution operation. 13

Figure 2.7. Simple max pooling operation. 14

Figure 2.8. Word2vec implementation architecture overview. 14

Figure 3.1. Sample Biocreative entity corpus. 21

Figure 3.2. Sample Biocreative relation corpus. 21

Figure 3.3. Example one hot representation of the word “right”. 22

Figure 3.4. Steps of obtaining candidate relations from raw abstract data. . . 23

Figure 3.5. Sample input representation. 25

Figure 3.6. Sample sentence image. 26

Figure 3.7. BiLSTM model that is used in the thesis work. 28

Figure 3.8. CNN model that is used in the thesis work. 31

Figure 3.9. System overview. 32

LIST OF TABLES

Table 2.1. Best scores for each team in BioCreative VI task ChemProt. 18

Table 3.1. Distribution of Biocreative dataset. 20

Table 3.2. List of part of speech tags. 24

Table 3.3. BiLSTM model hyperparameters. 28

Table 3.4. CNN model hyperparameters. 30

Table 4.1. Computing environment. 36

Table 4.2. Word2vec network parameters. 37

Table 4.3. Results of word embedding comparison with BiLSTM. 38

Table 4.4. BiLSTM model parameters used in word embedding comparison. . 38

Table 4.5. CNN model parameters used in word embedding comparison. . . . 39

Table 4.6. Results of word embedding comparison with CNN. 40

Table 4.7. Results of trainable vs. fixed word embedding comparison with BiLSTM. 41

Table 4.8. BiLSTM parameters in trainable vs. fixed word embedding comparison. 42

Table 4.9. CNN parameters in trainable vs. fixed word embedding comparison. 42

Table 4.10. Results of trainable vs. fixed word embedding comparison with CNN. 43

Table 4.11. Selected BiLSTM grid search results for position embedding. . . . 44

Table 4.12. BiLSTM grid search space for position embedding experiments. . . 44

Table 4.13. Results of position embedding experiments with BiLSTM. 45

Table 4.14. Selected CNN grid search results for position embedding. 46

Table 4.15. CNN grid search space for position embedding experiments. 46

Table 4.16. Results of position embedding experiments with CNN. 47

Table 4.17. Results of POS tag embedding experiments with BiLSTM. 48

Table 4.18. Selected BiLSTM grid search results for POS tag embedding. . . . 49

Table 4.19. BiLSTM grid search space for POS tag embedding experiments. . . 49

Table 4.20. CNN grid search space for POS tag embedding experiments. 50

Table 4.21. Results of POS tag embedding experiments with CNN. 50

Table 4.22. Selected results of the CNN grid search for POS tag embedding. . 51

Table 4.23. Results of IOB tag embedding experiments with BiLSTM. 52

Table 4.24. BiLSTM grid search space for IOB tag embedding experiments. . . 52

Table 4.25. Selected BiLSTM grid search results for IOB embedding. 53

Table 4.26. Selected CNN grid search results for IOB embedding. 53

Table 4.27. CNN grid search space for IOB tag embedding experiments. 54

Table 4.28. Results of IOB tag embedding experiments with CNN. 54

Table 4.29. Evaluation of selected best models on the test set. 56

Table 4.30. Average precision of selected models on the test set. 56

Table 4.31. The selected best BiLSTM model. 56

Table 4.32. The selected best CNN model. 57

LIST OF SYMBOLS

b_p	Bias value of perceptron unit p
c_t	State value of LSTM unit at time t
f_t	Forget gate value of LSTM unit at time t
h_t	Output value of LSTM unit at time t
i_t	Input gate value of LSTM unit at time t
k_d	Dissociation constant
k_i	Inhibition constant
o_t	Output gate value of LSTM unit at time t
v_{hi}	Weight from input node i to hidden unit h
w_{jh}	Weight from hidden unit h to output unit j
x_p	Input value of perceptron p
y_p	Output value of perceptron p
z_h	Value of hidden unit h in word2vec
σ	Sigmoid function

LIST OF ACRONYMS/ABBREVIATIONS

ANN	Artificial neural network
BiLSTM	Bidirectional long short term memory
CNN	Convolutional neural network
CRF	Conditional random fields
EC50	Half maximal effective concentration
GAN	Generative adversarial network
IC50	Half maximal inhibitory concentration
IOB	Inside outside beginning
LSTM	Long short term memory
ML	Machine learning
MLP	Multilayered perceptron
NLP	Natural language processing
NMR	Nuclear magnetic resonance spectroscopy
POS	Part of speech
ReLU	Rectified linear unit
RNN	Recurrent neural network
Tanh	Hyperbolic tangent

1. INTRODUCTION

Knowledge regarding the interactions between ligands and proteins is indispensable for biomedical research and drug discovery. The knowledge is extracted from published articles in biomedical domain manually and then stored in structural databases. The process of extracting protein-ligand interactions from the biomedical literature is getting difficult because of the increasing number of published biomedical articles. Figure 1.1 depicts the number of indexed articles by Medline [1]. As observed in Figure 1.1, there is an increasing trend in the number of articles that are indexed by Medline for the previous 25 years. The situation leads to a need for an automatic way of extracting relations from biomedical literature. The aim of this thesis is to create deep learning models for biomedical relation extractions and analyze the effects of different linguistic features.

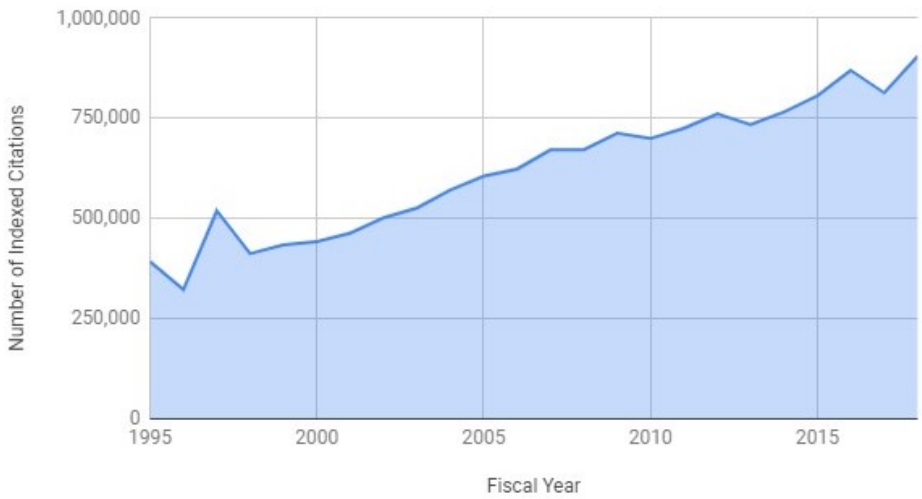

Figure 1.1. Number of indexed articles versus Fiscal year

Currently, there are various databases that provide information about protein-ligand interactions in a structural format: BindingDB, PDBbind, BindingMOAD [2–5]. These systems extract the interactions manually from biomedical literature. They scan the articles from selected journals, extract relations and then add the relations to the system. The current databases can only cover selected journals. In order to increase

the range of scanning, an automated system is required. Deep learning models can be trained to recognize these interactions from raw biomedical text. This thesis focuses on finding the optimal deep learning method with the optimal input representation. BiLSTM and CNN are selected as our deep learning models. There are various features used in the field of Natural Language Processing (NLP), some features are more useful than the others for specific tasks. Word embeddings, distance embeddings, POS tag embeddings and IOB chunk tag embeddings are used as features. Various experiments are conducted in order to compare the models.

2. BACKGROUND

2.1. Protein Ligand Interaction

Ligands and proteins are chemical compounds that satisfy predefined rules. Protein is a macromolecule that is a sequence of amino acids and exists in organisms. Amino acid is an organic compound. Chemicals that contain Carbon element are called organic compound. Amino acid should contain Amine and Carboxyl group attached to Carbon element. Our focus is Proteinogenic amino acids which means protein creating. There are over 500 amino acids that can be found in nature. However, 20 of them are directly encoded by human genome. Core structure of an amino acid can be seen in the following Figure 2.1. Left part of α-carbon is amine group which is (-NH_2) and right part of α-carbon is carboxyl group which is (-COOH). R represents the side chains in the amino acid.

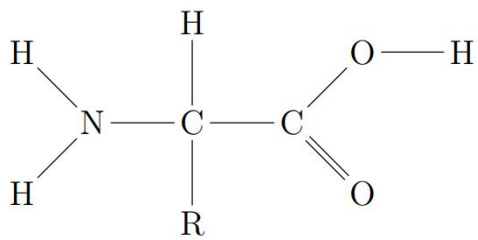

Figure 2.1. Core structure of an amino acid.

Ligand is a small compound that binds to a protein with non-covalent bond (intermolecule bonds). The idea of covalent and non-covalent bonding is directly related to electronegativity and basic structure of an atom. An atom consists of a nuclei and at least one electron attached to it. There is at least one proton and neutron in the nuclei. Atoms try to fill their last energy level and want to complete their farthest energy level with 8 electrons. There are different orbitals defined in the atom structure: s, p, d, f. S orbital holds 2 electrons and p orbital holds 6 electrons. Atoms share electrons in order to fill their last orbital and this sharing is called as covalent bond which

is interatomic binding. Electronegativity is a property of an atom that defines how much the atom wants to obtain an electron. Electronegativity is proportional to the number of valence electrons and distance of the last orbital to nuclei thus Fluorine is the most electronegative element. Non-covalent bond is a binding between molecules not between atoms. For example; water molecule (H_2O) is formed with a polar covalent bond. Oxygen and Hydrogen atoms share their atoms to fill their last energy level. However, Oxygen is much more electronegative than Hydrogen thus it pulls electrons to its nuclei and makes itself positive charged and makes Hydrogen negative charged thus this binding is called as polar covalent bond. When multiple water molecules come together, they bind to each other because of these polarized shape of water molecule; negative charged Hydrogen atoms are connected to positive charged Oxygen atoms because of electrostatic forces and we call it Hydrogen bond.

Proteins and ligands bind with intermolecule bonds which do not contain electron sharing. There are some covalent bonds that occur between proteins and ligands but they are not common. Proteins play many critical roles in human body and their function is mostly related to their three dimensional structure [6]. There are two methods to obtain the structure of a protein: Nuclear Magnetic Resonance Spectroscopy (NMR) and X-Ray Crystallography [7]. The conformation of a protein changes with binding of a ligand. Protein ligand binding is crucial for human body thus the binding mechanism and the structure of the protein attracts many researchers from different disciplines. For example, hemoglobin is a protein in red blood cells and it transports Oxygen molecules across the body. In this case, Oxygen molecule can be counted as a ligand and hemoglobin is a protein. The previous example emphasizes the importance of protein ligand interaction in living organisms.

Protein's function is highly related to its conformation. Conformation is the spatial arrangements of atoms in the molecule. In order to make this idea clearer, ethane molecule can be used as an example. Ethane molecule consists of 2 Carbon atoms and 6 Hydrogen atoms (H_3C-CH_3). Each dihedral angle corresponds to a different conformation of ethane. Potential energy of the molecule fluctuates between [3,0] kcal/mol.

The potential energy of a molecule is an important concept because it is used to calculate binding affinity of a protein ligand complex in a theoretical way by computers. There are two specific conformations of ethane: eclipsed and staggered. Both of the conformations are depicted in Figure 2.2. The change in energy is caused because of electrostatic forces. Hydrogen atoms are positive charged because of polar covalent bond. The molecule has more potential energy in eclipse formation because Hydrogen atoms are closer to each other and the cause of potential energy is called torsional strain or eclipsing strain. Conformation of a molecule is important to calculate the

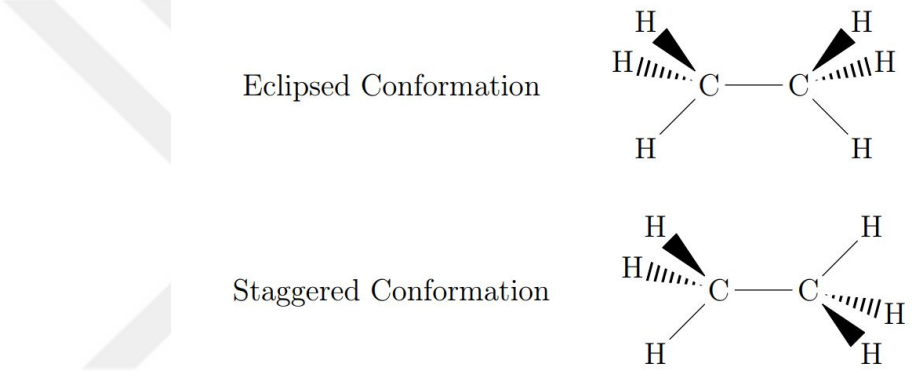

Figure 2.2. Eclipsed and staggered conformations of ethane.

potential energy of the molecule and potential energy is used to calculate protein ligand binding affinity. We can calculate free binding energy by first-principles quantum mechanical simulation by treating each molecule explicitly in the solvent. Also, we can choose an easier path by treating water molecules implicitly and express them with hydrophobic effect and dielectric screening terms in the energy equation. The energy equation contains terms that represent energy caused by bond stretching, angle bending, torsion strain, improper dihedral, Van der Waals interactions and Coulombic interactions [8]. Protein ligand affinities can be measured experimentally and there are four major metrics: Dissociation constant (k_d), Inhibition constant (k_i), Half maximal effective concentration (EC50) and Half maximal inhibitory concentration (IC50). k_d is the ratio of ligand concentration to the complex concentration, similarly k_i is the ratio of inhibitor concentration to the complex concentration. IC50 shows how much ligand is needed to inhibit 50 percent of protein's activity. EC50 is opposite of IC50, it measure how much ligand is needed to stimulate protein's activity by 50 percent.

2.2. Protein Ligand Interaction Databases

There are three main databases that provide protein-ligand interaction data: PDPBind, BindingDB, BindingMOAD.

2.2.1. BindingDB

BindingDB is a publicly accessible protein ligand interaction database that is maintained by Skaggs School of Pharmacy and Pharmaceutical Sciences UC San Diego [2]. This database provides affinity data and experiment conditions (pH, temperature etc.) to medicinal chemists, computational chemists, and pharmacologists. BindingDB collects data from literature directly and continuously. Each BindingDB data entry contains one protein target, one compound, affinity values and information of the publication. If there are available experimental conditions, they are also included in the database. BindingDB collects data from patents too which is not included in other databases. All data collection is done manually by BindingDB staff. BindingDB scans the following scientific journals.

- ACS Chemical Biology
- ChemBioChem
- Chemical Biology and Drug Design
- Chemistry and Biology
- Nature Chemical Biology
- ACS BioChemistry
- Bioorganic Chemistry
- Journal of Biological Chemistry

BindingDB also imports data from PubChem, ChEMBL, PDSP Ki and CSAR databases. Another source of BindingDB is users, the system allows users to add experimental data to the database. After binding data are collected from the original source, another BindingDB staff checks the data in order to increase the reliability of

the dataset. BindingDB also sends emails to the authors of the original publications to check data in the database. Moreover, the system contains binding data that are obtained by using computational modelling. BindingDB is different from other major databases: PDBBind and BindingMOAD. PDBBind and BindingMOAD contain binding affinities of proteins that exist in PDB but BindingDB does not restrict the database to PDB. BindingDB contains hyperlinks to the following databases.

- PubMed and US Patents online.
- UniProt, BindingMOAD and DrugBank for protein details.
- ChEMBL, PubChem, UniChem and ZINC for compound details.
- AntibodyPedia for antibodies against protein targets.
- Reactome for biomolecular pathway of proteins.

In addition, the crystal structures in PDB are linked to BindingDB for detailed binding information. This link is valid in the reverse direction as well. BindingDB provides data in table format, each row corresponds to a protein-ligand complex. The table contains the following columns;

- Target information
- Ligand information
- Links to protein information
- Links to ligand information
- Link to complex information
- Experiment data

BindingDB provides various ways to search the database. First way is to search via proteins which is text-based search. The database also allows to search via compounds. The system has two main pages about compounds: FDA-approved products page and important compounds pages. Users can search compounds by entering SMILES or InChI strings. Also, users can search via article, patent, author, institution data.

2.2.2. PDBbind

PDBbind is another protein-ligand interaction database that is maintained by Shanghai Institute of Organic Chemistry under mutual agreement with University of Michigan. PDBbind starts to obtain data from PDB unlike BindingDB. First, it obtains the complex data from PDB. PDBbind developed its own classification scheme on a given structural information. There are four valid complexes: (i) complexes formed between protein and small-molecule ligand, (ii) complexes formed between nucleic acid and small-molecule ligand, (iii) complexes formed between two protein molecules and (iv) complexes formed between protein and nucleic acid. In order to distinguish valid organic ligands from other molecules, PDBbind uses a dictionary (Chemical Component Dictionary) that is also provided by PDB. PDBbind defines protein-protein complex as at least two peptide chain from different proteins.

After obtaining biomolecular complexes from PDB, PDBbind collects affinity data from literature. Each PDB structure file contains a "primary reference" data which is the referenced publication. PDBbind starts with scanning this article. In order to obtain corresponding experimental settings and binding assay method, they use a script that scans an article and look for keywords provided by PDBbind. PDBbind looks for three major measures for binding affinity: dissociation constant K_d, inhibition constant K_i and concentration at 50% inhibition (EC50). One of the motivations of PDBbind is to provide binding data for molecular docking and structure-based drug design. Not all entries in the database are complete, some entries lack information thus PDBbind provides a smaller "refined set" for designing and validating molecular docking and scoring methods. There are rules that are applied when preparing "refined set", the current rules are described in detail in [3].

The database is updated in annual basis with respect to PDB. In the first week of new year, entire PDB is downloaded and processed. Binding data is increased around 15% each year. The database is used for various research areas. For example; AutoDock used PDBbind in their system. AutoDock is a molecular docking and virtual

screening program created by Scripps Research [9]. PDBbind also developed CASF benchmark [3]. This benchmark aims to provide a evaluation of scoring functions [10]. The system is also used for structure based drug design [11].

2.2.3. BindingMOAD

BindingMOAD is another binding database that is created by Carlson Lab at University of Michigan [4, 5]. BindingMOAD use top-down approach to curate the dataset. The term "top-down" approach is a term they use in their article to describe their method. They choose to start with protein-ligand complexes that have 3D structure in PDB. They start with PDB search like PDBbind database. After they obtain valid protein-ligand structures, they read the literature for that complex, protein and ligand. Their aim is to validate their analysis on the complex and get the affinity data from the article. BindingMOAD also creates functional groups to compare related systems.

2.3. Deep Learning

Deep Learning is a branch of Machine Learning (ML) that is based on Artificial Neural Network (ANN). Artificial neural networks are inspired by biological neural networks and each neuron is called a perceptron in artificial neural networks.

The idea of the perceptron is proposed by Frank Rosenblatt in 1958 [12]. A single perceptron is depicted in Figure 2.3 and mathematical representation of a perceptron is $y_p = W_p^T x_p + b_p$ where W_p is the weight tensor of the unit, x_p is the input tensor to the unit, b_p is the bias tensor of the unit and y_p is the output of the unit. A single perceptron is not able to solve basic problems. For example, a single perceptron can not represent a XOR gate. The problem is demonstrated by Minsky and Papert in a book titled "Perceptrons: an introduction to computational geometry" [13]. Perceptrons are powerful when they are connected to each other thus Multilayered Perceptron (MLP) architecture is required to solve complex problems. Training a MLP requires backprop-

agation algorithm which is proposed by David E. Rumelhart, Geoffrey E. Hinton and Ronald J. Williams [14]. Intuition of the backpropagation algorithm is to propagate the error in the output layer to the input layer. With the backpropagation algorithm, number of research on artificial neural networks increased rapidly. However, in order to make MLP effective, non-linearity should be applied in each layer. Otherwise, adding layer to the network will be ineffective. Non-linear functions are applied to the output of the perceptron units and these functions are called activation functions. There are various activation functions currently used in the field of deep learning such as Rectified Linear Unit (ReLU), Sigmoid and Hyperbolic Tangent (Tanh).

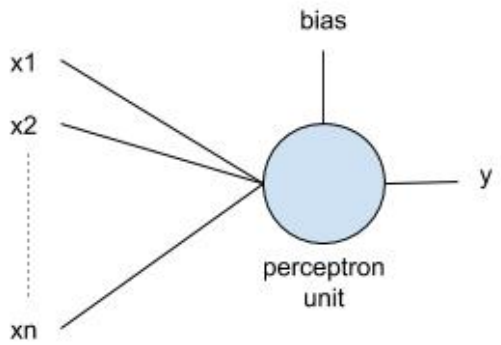

Figure 2.3. Example perceptron unit.

Today, we have various neural network approaches: Recurrent Neural Network (RNN), Convolutional Neural Network (CNN) and Generative Adversarial Network (GAN). RNN suffers from exploding and vanishing gradient problem, the problem is explored by Sepp Hochreiter in his master thesis [15]. Long Short Term Memory is proposed to solve the problem by Sepp Hochreiter [16]. Long Short Term Memory (LSTM) and Convolutional Neural Network are selected as our models in this study.

2.3.1. Long Short Term Memory

Long Short Term Memory (LSTM) is proposed to solve vanishing and exploding gradient problem in recurrent neural networks. The idea of LSTM is to adjust perceptron outputs via gates. The first proposed LSTM cell consists of two gates: input and output gate [16]. Figure 2.4 depicts the first LSTM cell with two gates. Each gate and input unit takes the same input: x_t and h_{t-1}. The input is represented by x_t and previous value of LSTM cell is represented by h_{t-1}.

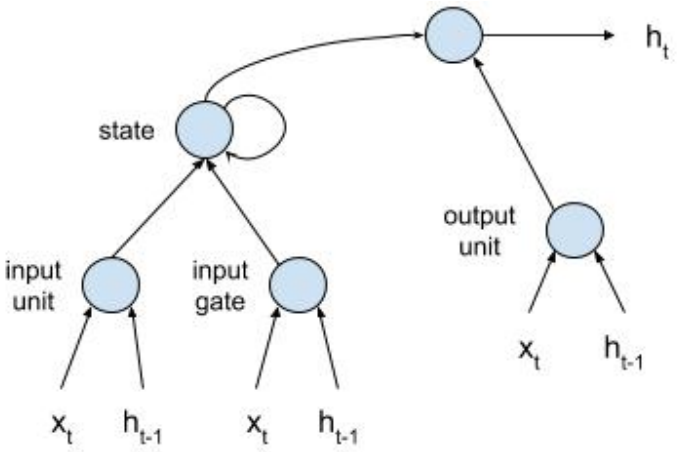

Figure 2.4. LSTM cell with two gates.

In 1999, a new gate is proposed to the initial LSTM cell that is proposed by Felix Gers *et al.* [17]. The LSTM cell with three gates is depicted in Figure 2.5. Currently, there are various implementations of LSTM cell based on different priorities. For example, there are LSTM block cell, LSTM block fused cell, LSTM cell with peephole in TensorFlow API provided by Google [18]. LSTM block cell is implemented based on work of Zaremba *et al.* [19]. LSTM cell with peephole is implemented based on works of Sak *et al.* [20]. The mathematical formulation for a LSTM cell is shown in Equations from 2.1 to 2.5.

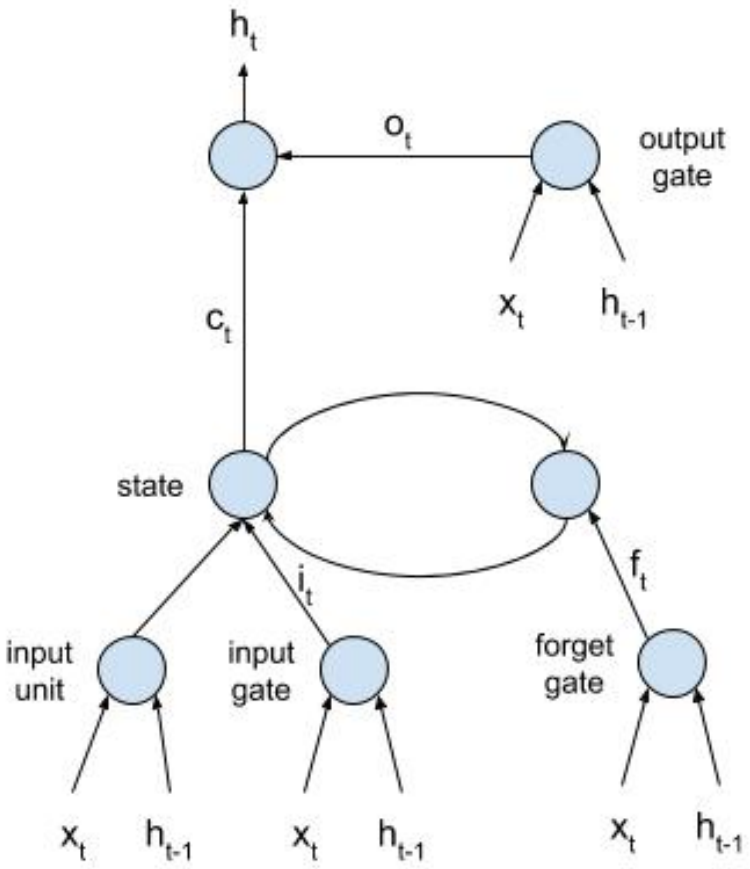

Figure 2.5. LSTM cell with three gates.

$$f_t = \sigma(W_f \cdot x_t + V_f \cdot h_{t-1} + b_f) \tag{2.1}$$

$$i_t = \sigma(W_i \cdot x_t + V_i \cdot h_{t-1} + b_i) \tag{2.2}$$

$$o_t = \sigma(W_o \cdot x_t + V_o \cdot h_{t-1} + b_o) \tag{2.3}$$

$$c_t = f_t \cdot c_{t-1} + i_t \cdot \sigma(W_c \cdot x_t + V_c \cdot h_{t-1} + b_c) \tag{2.4}$$

$$h_t = o_t \cdot \sigma(c_t) \tag{2.5}$$

2.3.2. Convolutional Neural Network

Convolutional Neural Network (CNN) takes a different approach than conventional neural network architectures. Convolution operation is what makes CNN different from other approaches. The operation is mainly used in the area of signal processing in order to modify a signal with respect to a filter. The operation is basically moving a signal over another signal and the moving signal is called filter or kernel. The mathematical representation of convolution operation is Equation 2.6.

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau \tag{2.6}$$

The convolution operation in CNN is similar to the operation in signal processing, we select a filter tensor and move the filter tensor over our input tensor. We select filter size as hyperparameter that is not optimized by the network. Our filter tensor represents weights and the network tries to optimize these weights. The intuitive approach to CNN is that each filter is optimized to recognize a unique shape. For example, if the input consists of face photos, then the filter should be optimized to recognize nose and some other filter should be optimized to recognize eyes. Modern CNN architectures are proposed by Yann LeCun *et al.* [21] and these neural networks called LeNet in the article. LeNet consists of convolution and max pooling layers. Convolution layer applies the selected number of filters to the input and returns a new tensors.

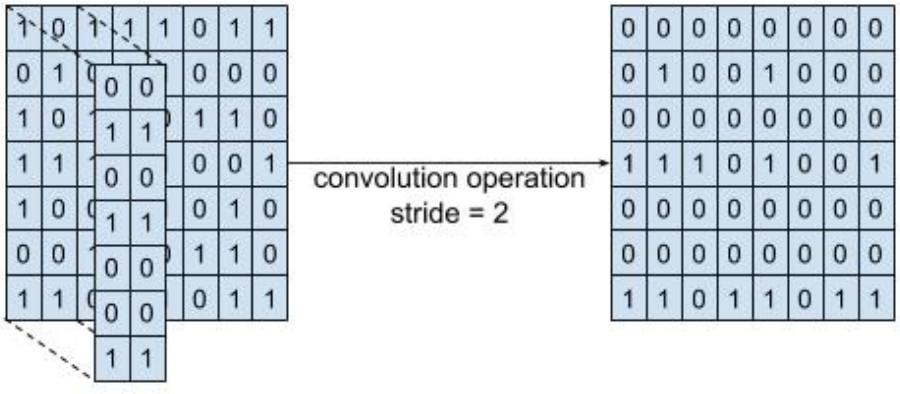

Figure 2.6. Simple convolution operation.

Convolution operation in CNN is depicted in Figure 2.6. After convolution layer, we need to apply subsampling method and the method should be non-linear in order to be useful in the neural network. LeNet uses max pooling operation which is still mostly used subsampling operation in CNN architectures. Max pooling operation is a non-linear operation that aims to subsample the input tensor. A filter size is selected and the selected filter moves over the input, selects the maximum value in the area of filter. Max pooling operation is depicted in Figure 2.7.

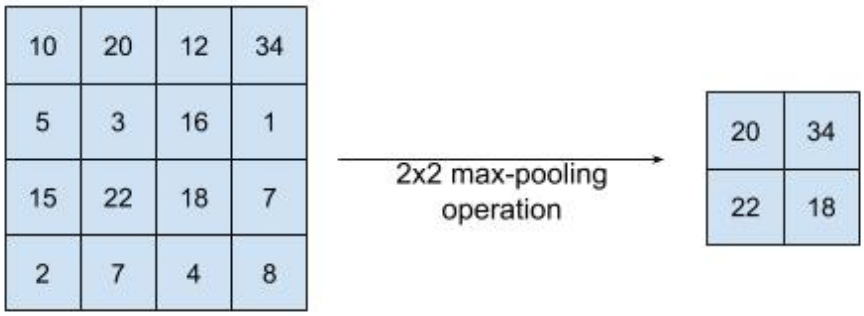

Figure 2.7. Simple max pooling operation.

2.3.3. Word2Vec

Word2vec algorithm is basically training a Multilayered Perceptron to predict words that occur in a selected window with the input token. The weights in the network are used as word embeddings. The overview of the network is shown in Figure 2.8. There are three layers in the architecture: input, hidden and output.

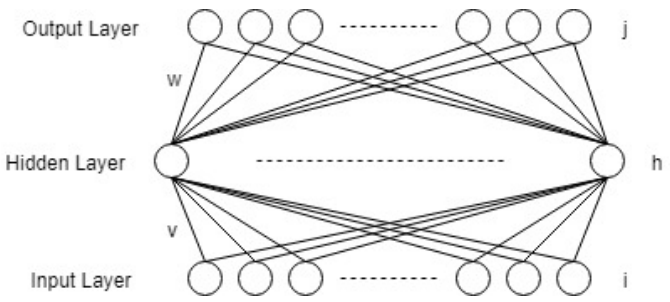

Figure 2.8. Word2vec implementation architecture overview.

Input layer represents the input values, hidden layer applies dot product to the input with hidden layer weights and we apply two operations in the output layer: dot product and softmax function. Dynamics of the network is described in Equations 2.7, 2.8, and 2.9.

$$z_h = v_h^T x \tag{2.7}$$

$$o_j^t = \sum_{h=1}^{H} w_{jh} z_h^t + w_{j0} \tag{2.8}$$

$$y_j^t = \frac{exp(o_j^t)}{\sum_k exp(o_k^t)} \tag{2.9}$$

The equations 2.8 and 2.9 are applied in the output layer. Equation 2.8 is dot product in the output layer and Equation 2.9 is the softmax function. In the hidden layer, we do not apply any activation function.

Output layer consists of all words in dictionary thus applying a softmax operation in output layer is a computationally heavy operation. Hiererchical softmax is proposed to increase performance of the output layer. Error function is cross entropy which is Equation 2.10.

$$E(W, V|X) = -\sum_t \sum_i r_i^t log y_i^t \tag{2.10}$$

We use the backpropagation algorithm to update the weights in the network. We take derivation of error function with respect to the weights and obtain update functions. There are two different set of weights in the network: hidden layer weights and output layer weights. Hidden layer weights are shown with v_{ih} where i represents input index and h represents hidden unit index. Output layer weights are shown with w_{jh} where j is output unit index and h is hidden unit index.

First, we derive the error function with respect to the output layer weights which is w_{jh} and obtain the update function for w_{jh}. Our update function for w_{jh} is

$$\Delta w_{jh} = -\mu \frac{\partial E}{\partial w_{jh}} \tag{2.11}$$

$\partial E/\partial w_{jh}$ can be written as Equation 2.12 using chain rule.

$$\frac{\partial E}{\partial w_{jh}} = \frac{\partial E}{\partial y_j} \cdot \frac{\partial y_j}{\partial o_j} \cdot \frac{\partial o_j}{\partial w_{jh}} + \sum_{k \neq j}^{J} \frac{\partial E}{\partial y_k} \cdot \frac{\partial y_k}{\partial o_j} \cdot \frac{\partial o_j}{\partial w_{jh}} \tag{2.12}$$

The required partial derivations are listed below:

$$\frac{\partial E}{\partial y_j} = -r_j \cdot \frac{1}{ln2} \cdot \frac{1}{y_j} \qquad \frac{\partial E}{\partial y_k} = -r_k \cdot \frac{1}{ln2} \cdot \frac{1}{y_k} \qquad \frac{\partial y_j}{\partial o_j} = y_j \cdot (1 - y_j) \tag{2.13}$$

$$\frac{\partial y_k}{\partial o_j} = -y_j \cdot y_k \qquad \frac{\partial o_j}{\partial w_j h} = z_h \tag{2.14}$$

We can obtain the Equation 2.15 via replacing Equation 2.12 with partial derivations in Equation 2.13 and 2.14.

$$\frac{\partial E}{\partial w_{jh}} = \Big[- r_j \cdot \frac{1}{ln2} \cdot \frac{1}{y_j} \cdot y_j \cdot (1 - y_j) \cdot z_h \Big] + \sum_{k \neq j}^{J} r_k \cdot \frac{1}{ln2} \cdot \frac{1}{y_k} \cdot y_k \cdot y_j \cdot z_h \tag{2.15}$$

Then we simplify Equation 2.15.

$$\begin{aligned}
\frac{\partial E}{\partial w_{jh}} &= \frac{1}{ln2} \Big[- \big(r_j \cdot \frac{1}{y_j} \cdot y_j \cdot (1 - y_j) \cdot z_h \big) + \sum_{k \neq j}^{J} r_k \cdot \frac{1}{y_k} \cdot y_k \cdot y_j \cdot z_h \Big] \\
&= \frac{1}{ln2} \Big[- \big(r_j \cdot (1 - y_j) \cdot z_h \big) + \sum_{k \neq j}^{J} r_k \cdot y_j \cdot z_h \Big] \\
&= \frac{1}{ln2} \Big[- r_j \cdot z_h + \Big(z_h \cdot y_j \big(r_j + \sum_{k \neq j}^{J} r_k \big) \Big) \Big] \\
&= \frac{1}{ln2} \cdot z_h \cdot \big(y_j - r_j \big)
\end{aligned}$$

We could write update function for w_{jh} as in Equation 2.16.

$$\Delta w_{ih} = -\mu \cdot \frac{1}{ln2} \sum_{t} z_h^t \cdot \left(y_j^t - r_j^t\right) \tag{2.16}$$

After obtaining update function for w_{jh}, it is easier to obtain v_{hi}. We modify the Equation 2.12 and obtain Equation 2.17.

$$\frac{\partial E}{\partial v_{hi}} = \frac{\partial E}{\partial y_j} \cdot \frac{\partial y_j}{\partial o_j} \cdot \frac{\partial o_j}{\partial z_h} \cdot \frac{\partial z_h}{\partial v_{hi}} + \sum_{k \neq j}^{J} \frac{\partial E}{\partial y_k} \cdot \frac{\partial y_k}{\partial o_j} \cdot \frac{\partial o_j}{\partial z_h} \cdot \frac{\partial z_h}{\partial v_{hi}} \tag{2.17}$$

We calculate the required partial derivations as follows;

$$\frac{\partial E}{\partial y_j} = -r_j \cdot \frac{1}{ln2} \cdot \frac{1}{y_j} \qquad \frac{\partial E}{\partial y_k} = -r_k \cdot \frac{1}{ln2} \cdot \frac{1}{y_k} \qquad \frac{\partial y_j}{\partial o_j} = y_j \cdot (1 - y_j) \tag{2.18}$$

$$\frac{\partial y_k}{\partial o_j} = -y_j \cdot y_k \qquad \frac{\partial o_j}{\partial z_h} = w_{jh} \qquad \frac{\partial z_h}{\partial v_{hi}} = x_i \tag{2.19}$$

We replace Equation 2.17 with Equations 2.18 and 2.19 then obtain Equation 2.20.

$$\frac{\partial E}{\partial v_{hi}} = \Big[- r_j \cdot \frac{1}{ln2} \cdot \frac{1}{y_j} \cdot y_j \cdot (1 - y_j) \cdot w_{jh} \cdot x_i \Big] + \sum_{k \neq j}^{J} r_k \cdot \frac{1}{ln2} \cdot \frac{1}{y_k} \cdot y_k \cdot y_j \cdot w_{jh} \cdot x_i \tag{2.20}$$

We simplify Equation 2.20 in order to get clearer equations.

$$\begin{aligned}
\frac{\partial E}{\partial v_{hi}} &= \frac{1}{ln2} \Big[- \big(r_j \cdot \frac{1}{y_j} \cdot y_j \cdot (1 - y_j) \cdot w_{jh} \cdot x_i\big) + \sum_{k \neq j}^{J} r_k \cdot \frac{1}{y_k} \cdot y_k \cdot y_j \cdot w_{jh} \cdot x_i \Big] \\
&= \frac{1}{ln2} \Big[- r_j \cdot w_{jh} \cdot x_i + r_j \cdot w_{jh} \cdot x_i \cdot y_j + y_j \cdot w_{jh} \cdot x_i \sum_{k \neq j}^{J} r_k \Big] \\
&= \frac{1}{ln2} \Big[- r_j \cdot w_{jh} \cdot x_i + \Big(w_{jh} \cdot x_i \cdot y_j \big(r_j + \sum_{k \neq j}^{J} r_k\big) \Big) \Big] \\
&= \frac{1}{ln2} \cdot w_{jh} \cdot x_i \cdot \big(y_j - r_j\big)
\end{aligned}$$

We could write update function for v_ih in Equation 2.21.

$$\Delta v_{hi} = -\mu \cdot \frac{1}{ln2} \sum_t w_{jh}^t \cdot x_i^t \cdot \left(y_j^t - r_j^t\right) \tag{2.21}$$

2.4. Related Work

This thesis work aims to extract relations between chemicals and proteins from biomedical literature. Biocreative dataset is used in this research and it is obtained from Biocreative Challenge VI. Biocreative challenge aims to promote research in specific areas in the field of bioinformatics. In Biocreative VI, ChemProt task is organized in order to promote research in biomedical relation extraction. The aim in ChemProt task is to extract relations and label relations with different relation types.

Table 2.1. Best scores for each team in BioCreative VI task ChemProt.

Team ID	**Precision**	**Recall**	**f1-measure**
374 [22]	0.57	0.47	0.51
379 [23]	0.53	0.46	0.49
394 [24]	0.29	0.32	0.30
397 [25]	0.60	0.11	0.18
403 [26]	0.56	0.67	0.61
404 [27]	0.33	0.40	0.37
417 [28]	0.66	0.55	0.60
421	0.16	0.34	0.21
424 [29]	0.67	0.51	0.58
427 [30]	0.26	0.66	0.38
430 [31]	0.72	0.57	0.64
432 [32]	0.47	0.44	0.45
433 [33]	0.63	0.51	0.56

However, the focus in our study is to extract relations and label them as no relation or relation. Gold standard corpus is created for ChemProt task. Dataset is explained in Section 3.1 in detail. Thirteen teams participated in the task and each team was allowed to submit five runs at most [34]. The best f1-measure scores for each team are listed in Table 2.1.

Both conventional machine learning algorithms and deep learning algorithms are used in the task. The best f1-measure is obtained by work of Peng *et al.* [31]. The work is based on ensemble of three different algorithm: CNN, BiLSTM and SVM. In this thesis, for the CNN architecture, the work of Peng *et al.* [35] is used as base model and for the BiLSTM architecture, the work of Kavuluru *et al.* [36] is used as base model.

3. MATERIALS AND METHODS

3.1. Dataset

We used a dataset that is curated for BioCreative challenge. BioCreative is a contest that aims to assess the state of art methods for various tasks in bioinformatics domain. In BioCreative VI, chemical protein relation extraction is selected as a task in the contest. The corpus consists of three files: abstracts, relations and entities. Abstracts file contains raw abstract text and unique id that is assigned to each abstract. Entities file contains entity details and entity id that is assigned to each entity with respect to the abstract that entity occurs. Relations file contains relations between protein and chemical entities. Table 3.1 shows the distribution of BioCreative dataset [37]. CPR is an abbreviation for "Predicted chemical-Protein relation" which represents the relation type. Examples of entities are shown in Figure 3.1 and examples of relations are shown in Figure 3.2. There are six columns in the Figure 3.1: abstract id, entity id, entity type, start index, end index, and entity name. There are six columns in the Figure 3.2: abstract id, relation id, evaluation status, relation type, first argument id, and second argument id.

Table 3.1. Distribution of Biocreative dataset.

	Training	**Development**	**Test**
Document	1020	612	800
Chemical	13017	8004	10810
Protein	12752	7567	10019
CPR:3	768	550	665
CPR:4	2254	1094	1661
CPR:5	173	116	195
CPR:6	235	199	293
CPR:9	727	457	644

23538162	T1	CHEMICAL	1305	1308	Rg1
23538162	T2	CHEMICAL	291	306	Ginsenoside Rg1
23538162	T3	CHEMICAL	308	311	Rg1
23538162	T4	CHEMICAL	549	552	Rg1
23538162	T5	CHEMICAL	581	584	Rg1
23538162	T6	CHEMICAL	730	738	nitrogen

Figure 3.1. Sample Biocreative entity corpus.

23538162	CPR:4	Y	DOWNREGULATOR	Arg1:T5	Arg2:T19
23538162	CPR:4	Y	INDIRECT-DOWNREGULATOR	Arg1:T5	Arg2:T20
23538162	CPR:6	Y	ANTAGONIST	Arg1:T7	Arg2:T21
23538162	CPR:6	Y	ANTAGONIST	Arg1:T7	Arg2:T22
23538162	CPR:4	Y	INHIBITOR	Arg1:T8	Arg2:T23
23538162	CPR:6	Y	ANTAGONIST	Arg1:T10	Arg2:T24

Figure 3.2. Sample Biocreative relation corpus.

3.2. Input Representation

Input representation is a crucial aspect of the Natural Language Processing tasks because the data is in unstructured format. In order to use natural language as an input to our models we need to convert natural language data to quantitative data. There are two main ways to convert words to quantitative data: one-hot representation and word embeddings such as the ones obtained by the word2vec algorithm [38]. One-hot representation is a straightforward way to represent words. Let d be the number of words in our vocabulary then we create a vector with length d for each word. Each dimension of the vector represents an unique word in the vocabulary. We assign an unique id to each word and this id represents the dimension of the vector that corresponds to the word. Each word is represented with a vector where only one dimension is 1 and other dimensions are 0. An example one-hot representation is shown in Figure 3.3. One hot representation has two major drawbacks: sparse representation and no relation between words. Let d be dictionary size then $d-1$ values in the vector are 0 thus one hot representation is a sparse representation. The second drawback of the representation is that there is no any correlation between words. For example, “justice”

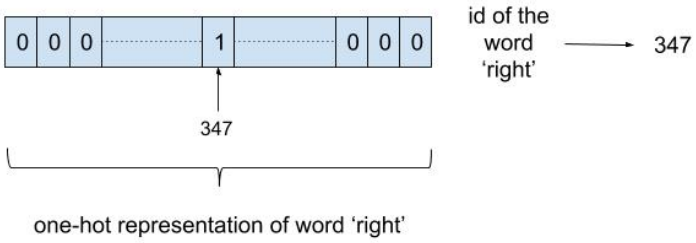

Figure 3.3. Example one hot representation of the word "right".

and "fairness" are words that are strongly correlated with each other because, these words refer to same idea. In one hot representation the semantic information is not encoded to the representation thus there is no any correlation between these two words in their representation. Word2vec algorithm solves these two problems, it creates a dense representation for each word and representations of words with semantic correlation are similar to each other. In our research, word2vec approach is selected to be used in our research. The next step is to create input from raw BioCreative dataset. Our raw dataset contains abstracts but abstracts should be converted to a suitable format in order to be used in relation extraction task. There are various ways to prepare input to the model. One way is to use whole abstract as an input, the other way is to split abstracts into sentences and use sentences as separate inputs to the model. The second approach is selected for data preparation. Sentences are split via NLTK [39].

After obtaining sentences, we have different ways to use the sentence as input. One way is to use whole sentence, the second way is to use text between entities in the sentence and the final option is to extend the text between entities with a window size w. The second approach is selected in our study. The text between entities is called candidate relation. In order to parse candidate relations, entities in each sentence are tagged. Entities are listed in the dataset and entity indexes are compared with sentence indexes in order to find entities in the sentence. The steps of processing candidate relations are depicted in Figure 3.4.

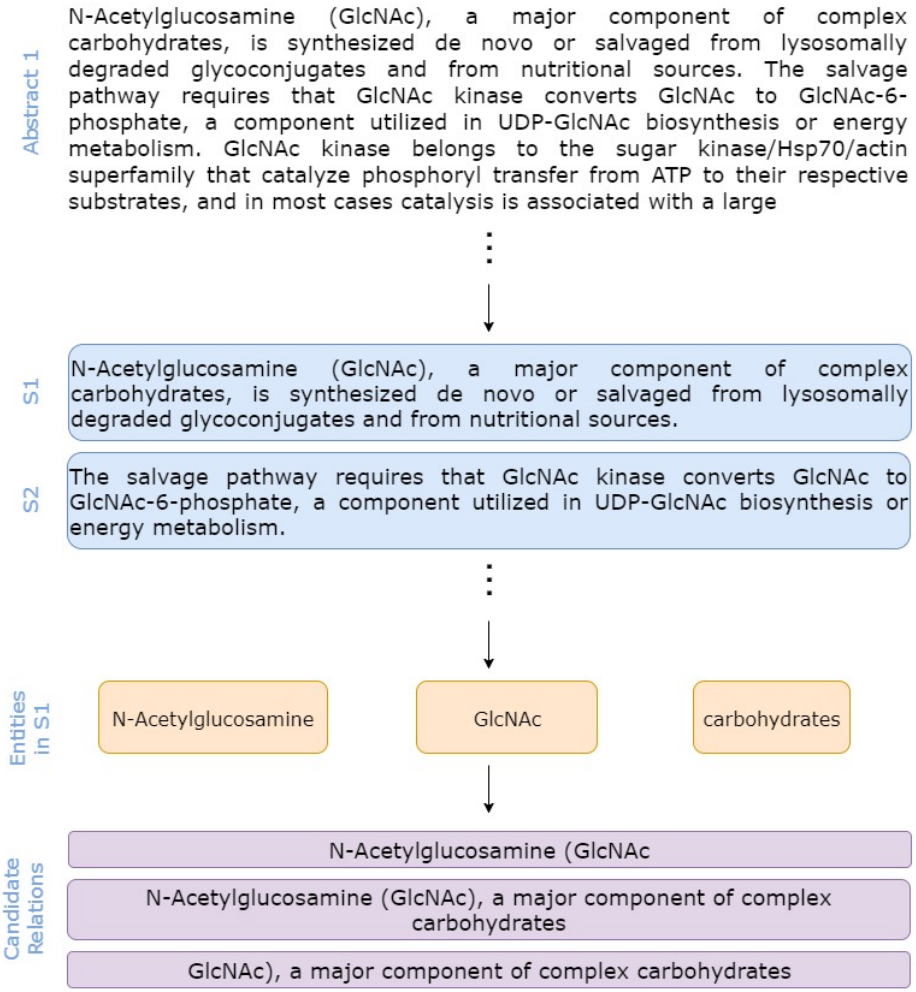

Figure 3.4. Steps of obtaining candidate relations from raw abstract data.

Candidate relation is sequence of words. NLTK is used to tokenize sentences and obtain tokens. Each token is mapped to a vector. The vector contains various information: word embedding, position embedding, POS tag embedding, IOB chunk embedding. Two types of word embeddings are used in this thesis. One is based on training the word2vec algorithm with biomedical text and is obtained from the work of Chiu *et al.* [40]. The other one is obtained by using Wikipedia articles as training set for word2vec. Position embeddings represents the distance of token to protein entity and the chemical entity. Part of speech tags are another information that is derived from text. NLTK is used to obtain part of speech tags and standard part of speech tags are listed in Table 3.2. Tokens can be combined and chunks can be created.

IOB is a way of representing chunks; I represents Inside, O represents Outside and B represents Beginning. Noun phrases (NP: <DT>? <JJ>* <NN>*), verb phrases (<V> <NP|PP>*) and prepositional phrases(<P> <NP>) are used for chucking, regular expression is used. The ? symbol matches if the preceding item occurs zero or one time and the * symbol matches if the preceding item occurs zero or more times. An example of prepared input is depicted in Figure 3.5.

Table 3.2. List of part of speech tags.

Label	**Explanation**
NN	Singular Noun
NNS	Plural Noun
NNP	Proper Noun
VBD	Past Tense Verb
VBZ	3rd Person Singular Present Tense Verb
VBP	Non 3rd Person Singular Present Tense Verb
VBN	Past Participle
PRP	Pronoun
PRP$	Possesive Pronoun
JJ	Adjective
IN	Preposition
DT	Determiner

For each candidate relation, a nxm matrix is created where n represents length of candidate relation and m represents embedding size. Each matrix is called sentence image and depicted in Figure 3.6. Each model is supplied with different tensors. BiLSTM requires three dimensional tensor consists of batch size, sentence length, and embedding size as input and CNN requires four dimensional tensor containing batch size, sentence length, embedding size, and channel number as input.

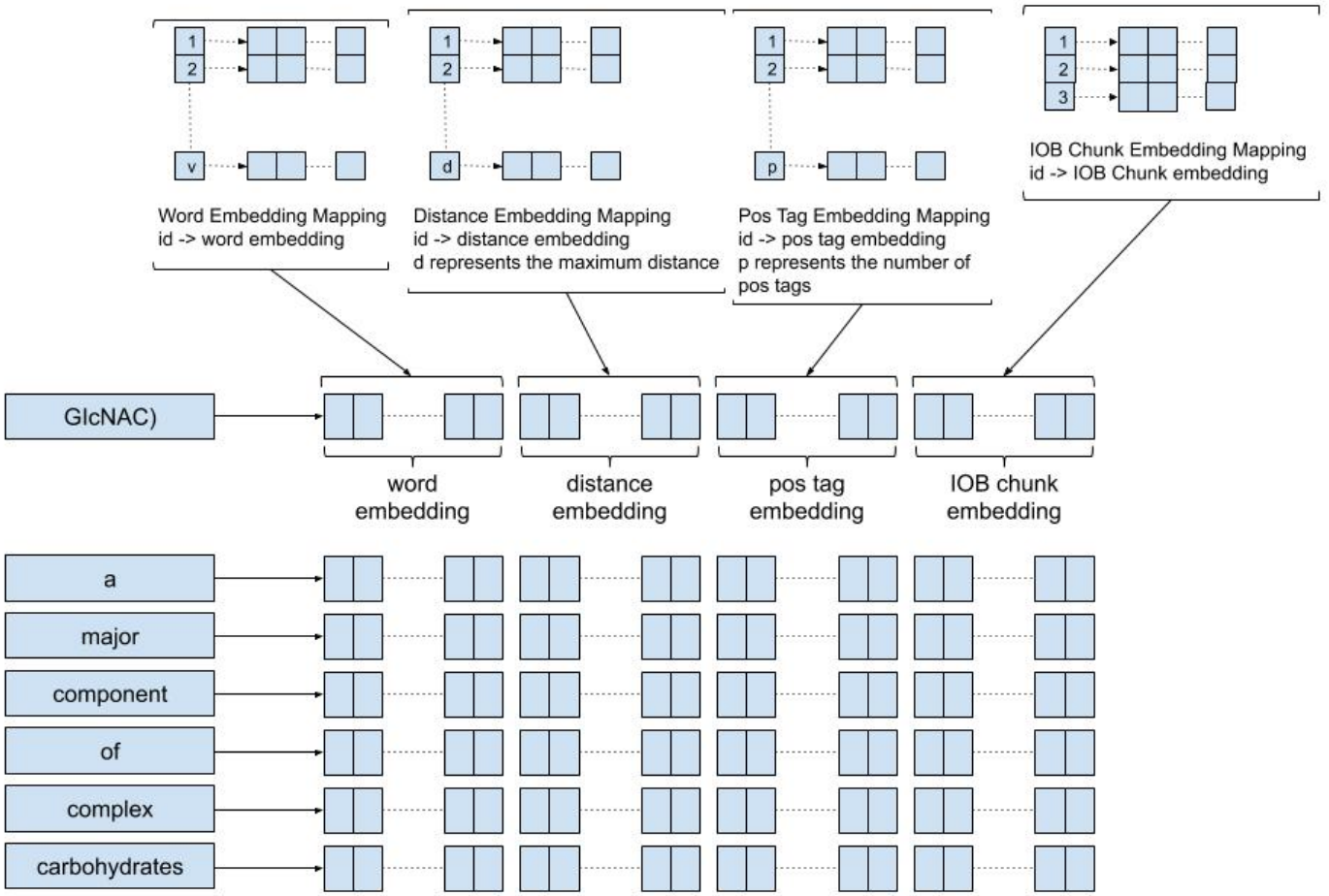

Figure 3.5. Sample input representation.

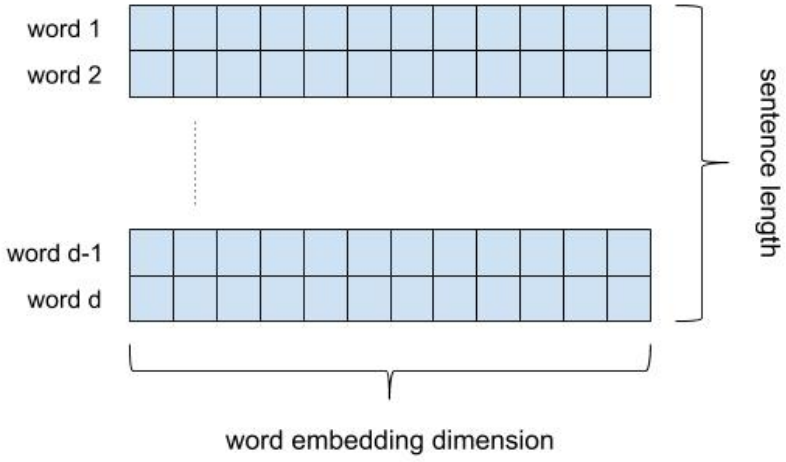

Figure 3.6. Sample sentence image.

3.3. Machine Learning Models

3.3.1. Bidirectional LSTM

Bidirectional LSTM is an enhanced version of recurrent neural networks. It is proposed to solve exploding and vanishing gradient problem of the RNN's [16]. BiLSTM is selected because of its success in natural language processing tasks.

BiLSTM is a neural network approach that consists of two independent LSTM units. Each LSTM unit is supplied with the same input sequence. However, their direction of processing the input is different. Internal dynamics of the LSTM unit is explained in Section 2.3.1 in detail. The idea of BiLSTM is merely combining two LSTM units and exploiting the input from both directions. BiLSTM architecture can also be combined with various different neural networks or conventional machine learning algorithms such as Conditional Random Fields (CRF). However, these complicated architectures can increase the need for computing power dramatically thus a plainer BiLSTM architecture is chosen in this research.

BiLSTM architecture is depicted in Figure 3.7. The architecture consists of three main layers: LSTM layer, max pooling layer, output layer. Firstly, the input is directly supplied to the LSTM layer. The LSTM layer consists of two main LSTM units and each unit has n number of hidden units. Each LSTM unit outputs n dimensional vector for each time t. The LSTM layer is followed by max pooling layer. Let T be the length of the input sequence and n be the number hidden units for each LSTM unit then max pooling layer is supplied with T times n dimensional vectors from each LSTM unit. Max pooling operation selects the maximum value of each dimension through the time. Max pooling layer outputs two n dimensional vectors. There is a concatenation operation between the max pooling layer and the output layer. The two n dimensional vectors are concatenated and supplied to the output layer. The output layer consists of two nodes, each representing two different outcomes: relation and no relation. The softmax function is applied to the output nodes in order to normalize the output value into the [0,1] interval. Softmax function for output node o_0 is depicted in Equation 3.1.

$$y_0^t = \frac{exp(o_0^t)}{exp(o_0^t) + exp(o_1^t)} \tag{3.1}$$

Cross entropy function is used to calculate the error between prediction and target value.

$$Error = \sum_t r_0 \cdot logy_0 + r_1 \cdot logy_1 \tag{3.2}$$

Dropout is applied in LSTM layer in order to prevent overfitting. Adam optimizer is used as an optimization algorithm in our model. Each model parameter is listed in Table 3.3. Some parameters are selected using the grid search algorithm and some parameters are selected as fixed parameters. In the optimal case, every hyperparameter should be selected using the grid search algorithm. However, because of the computational limitations only a subset of the hyperparameters are selected using the grid search algorithm.

Table 3.3. BiLSTM model hyperparameters.

Parameter Name	**Included in Grid Search**	**Value**
Batch Size	No	50
Number of Epoch	No	100
Learning Rate	No	0.001
Dropout Rate	No	0.5
Word Embedding Size	No	200
LSTM Hidden Units	Yes	128, 256, 512
Position Embedding Size	Yes	10, 20, 50, 100
POS Tag Embedding Size	Yes	10, 20, 50, 100
IOB Chunk Tag Embedding Size	Yes	10, 20, 50, 100

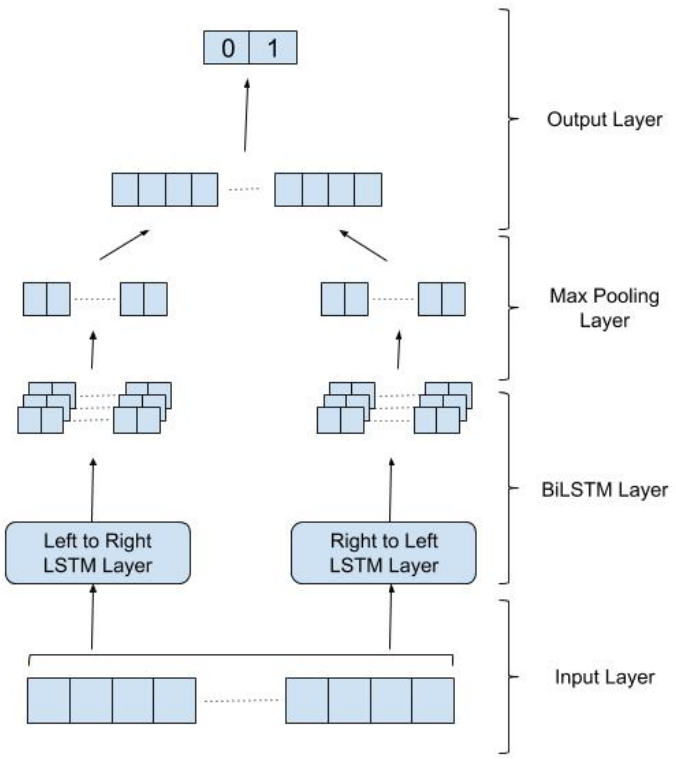

Figure 3.7. BiLSTM model that is used in the thesis work.

3.3.2. Convolutional Neural Network

The idea of Convolutional Neural Networks (CNN) is derived from the importance of localization in deep learning research. A CNN model does not process the whole input at one step, it divides the input into divisions and then processes each part independently. Finally, it combines the information that is obtained from each part. This approach is different from other neural network models thus CNN is selected as one of the predictor models in our research. CNN is also used in various natural language processing tasks. There are various CNN architectures in the literature. Various number of filters and convolution layers can be combined in the network. Also, different number of fully connected layers can be inserted before and after convolution layer. Work of Kim *et al.* is used as base CNN architecture in this thesis [41].

The CNN architecture is depicted in Figure 3.8. Our CNN model consists of three main layers: convolution layer, fully connected layer and output layer. Input is shaped into four dimensional tensor before it is supplied into CNN model, input preparation is discussed in Section 3.2. Firstly, the input is processed by convolutional layer. There are three different filter sizes in our convolutional layer. Let d be token total embedding size, then filter sizes are 2xd, 3xd and 4xd. Each filter aims to catch relation between 2, 3 and 4 tokens. Each filter outputs different vectors. Let n be sequence length, then the first filter (2xd) outputs a vector of size $(n-1)$x1. The second filter (3xd) outputs a vector of size $(n-2)$x1 and the third filter (4xd) outputs a vector of size $(n-3)$x1. After convolution layer, output vectors from each filter are concatenated and a single vector is created. The final vector is supplied into fully connected layer and finally fully connected layer is connected to output layer. Dropout layer is applied to the network in order to prevent overfitting. Relu is used as an activation function in fully connected layer. Softmax is used in output layer in order to normalize output values into interval of [0,1]. The softmax function is described in Equation 3.1. Cross entropy is used as error function in the model and it is shown in Equation 3.2. The hyperparameters of the CNN architecture are listed in Table 3.4. Batch size, number of epochs, learning rate, dropout rate and word embedding size are selected as fixed hyperparameters in

the model. CNN "filter out" parameter represents the number of filters in convolution layer for each filter size. For example, if CNN "filter out" parameter is selected 100 then there will be 300 filters in total. "FCL hidden unit size" represents the number of hidden units in the fully connected layer. These parameters are selected to be variable in grid search and selected optimal parameters for each test.

Table 3.4. CNN model hyperparameters.

Parameter Name	**Included in Grid Search**	**Value**
Batch Size	No	50
Number of Epoch	No	100
Learning Rate	No	0.001
Dropout Rate	No	0.5
Word Embedding Size	No	200
CNN Filter Out	Yes	150, 200, 250, 300, 350
FCL Hidden Unit Size	Yes	2048, 4096, 8192
Position Embedding Size	Yes	10, 20, 50, 100
POS Tag Embedding Size	Yes	10, 20, 50, 100
IOB Chunk Tag Embedding Size	Yes	10, 20, 50, 100

3.4. Implementation

Various experiments are conducted in this thesis thus a system for conducting experiments is implemented. The system that is used for experiments is depicted in Figure 3.9. There are four main components of the system: data processing, neural network models, predictor, and driver.

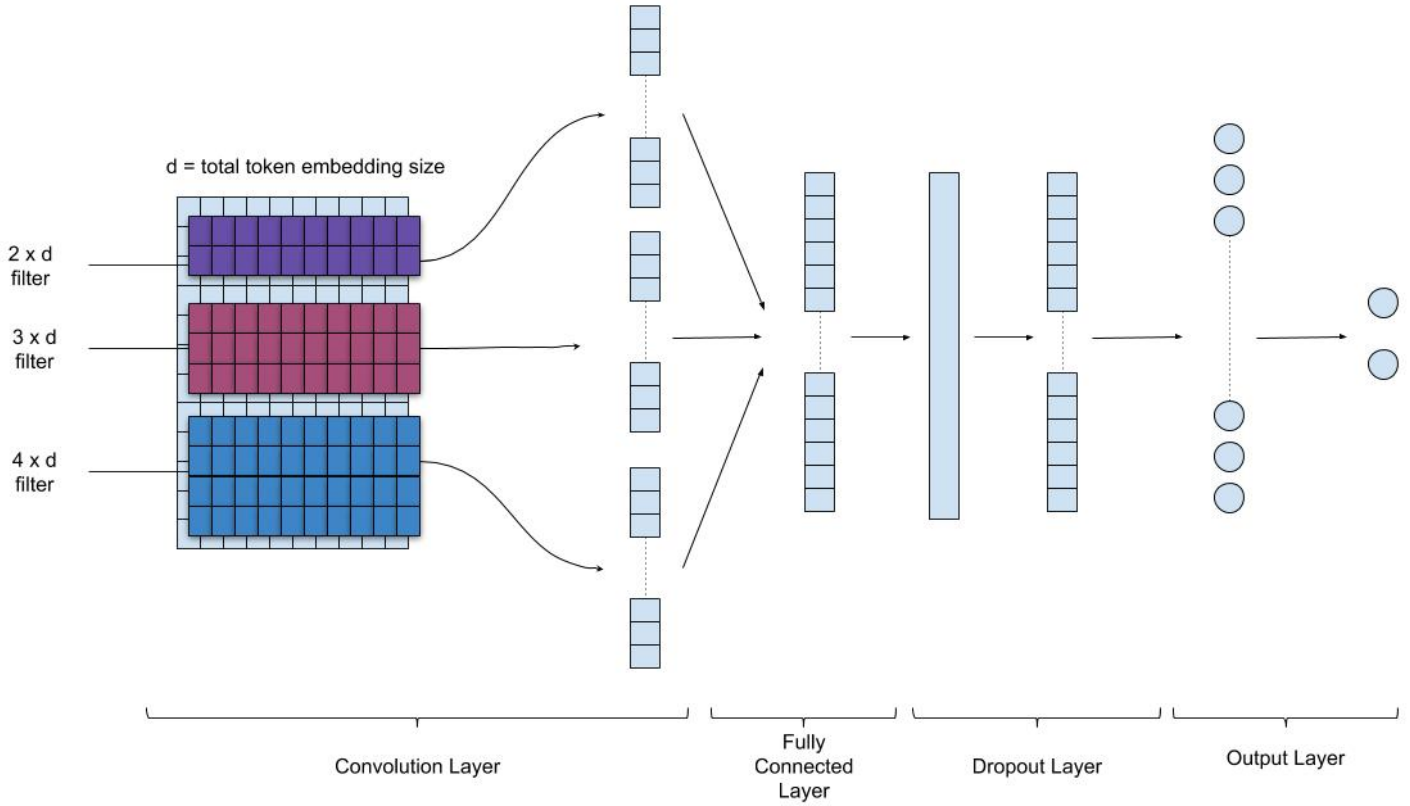

Figure 3.8. CNN model that is used in the thesis work.

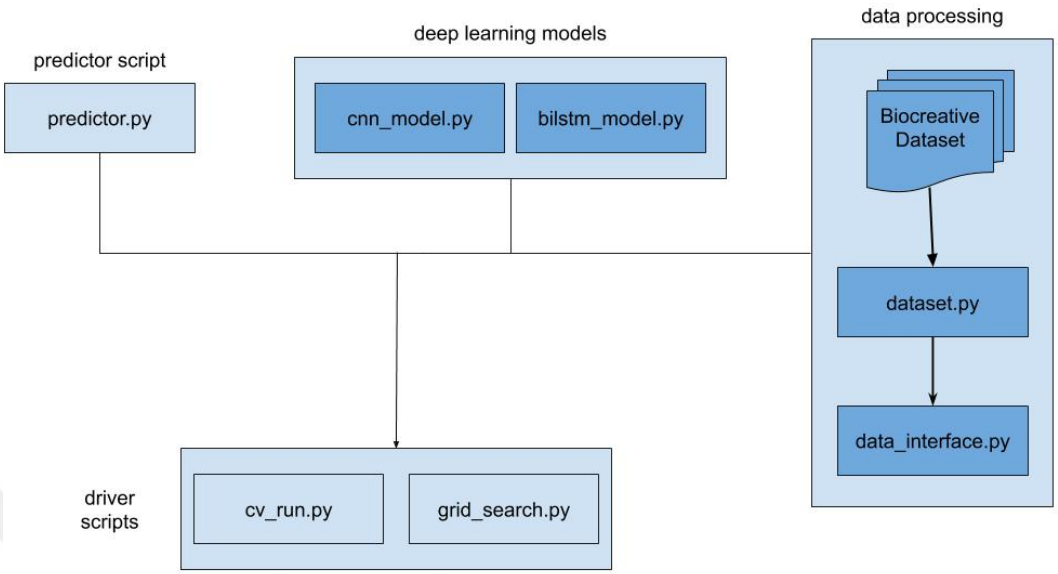

Figure 3.9. System overview.

The data processing component is responsible for reading a raw corpus, creating candidate relations, and providing data in batches for the models. The neural network model component is the implementation of the selected models via the Tensorflow library. The predictor script obtains data from the data component and runs the neural network models. The driver scripts combine the predictor script and the data then, run 5x2 cross validation paired t-test or grid search with selected parameters.

3.4.1. Data Processing Component

The data processing component is responsible for converting a raw corpus into a ready to use dataset. There are two classes in the component: dataset and data interface. The raw dataset files are processed by the dataset class and candidate relations are derived. The candidate relations are provided by the data interface in batches. Word embeddings, POS tag embeddings, IOB tag embeddings and position embeddings are also prepared by the data interface module.

3.4.1.1. Dataset Class. The dataset class contains two class methods and three static methods. The class methods are "prepare dataset" and "get dataset". The main method is the "prepare dataset" method which parses the raw corpus and creates candidate

relations. There are three files in the corpus and each of them is parsed by separate static methods. The label of the candidate relation is assigned by the static method "check instance relation".

3.4.1.2. Data Interface Class. The data interface class is responsible for processing candidate relations, creating feature embeddings and providing data in batches. Position embeddings, POS tag embeddings and IOB tag embeddings are initialized randomly via static methods. There are four methods that can be called from outside: "get batch", "get embedding information", "print information" and "write information".

3.4.2. Deep Learning Component

The deep learning component contains implementation of the BiLSTM and the CNN models with the Tensorflow library. Each model is implemented by separate classes. Each model class provides set of common methods such as optimize and prediction. There are also wrapper methods for simplicity.

3.4.2.1. BiLSTM model class. There are two helper methods in the BiLSTM model. One is used to initialize the BiLSTM layer and the other one is used to initialize the max pooling layer in the Tensorflow computation graph. There are two LSTM cells in the model, one processes data forward and the other one processes data backwards. There are also two dropout layers for each of the LSTM cells. LSTMBlockCell method is used for LSTM cell implementation from Tensorflow API. The method is implemented based on the work of Zaremba *et al.* [19]. There are various LSTM cells that are provided by the Tensorflow API. LSTMBlockCell is selected because it is a faster implementation. Static bidirectional RNN method is used in order to combine these two LSTM cells as BiLSTM. For the dropout implementation, dropout layer method is selected from the Tensorflow API. Each LSTM layer contains n hidden units and number of output units in the BiLSTM layer is $2n$. For each sequence, the max pooling operation selects the maximum value of each hidden unit through time. The prediction method is used to

create the part of the computational graph that is required for prediction and called only once through the life cycle of the module. The prediction part of the computational graph contains embedding layer, BiLSTM layer, and max pooling layer. The prediction method returns the node of the computational graph that calculates the values of the hidden units in the output layer. The optimize method is used to create the part of the computation graph that is required for optimization and called only once through the life cycle of the module. The method creates nodes for the loss function and Adam optimizer in the computational graph. The method returns the node for optimization.

3.4.2.2. CNN model class. The CNN model class has a similar structure to the BiLSTM model class. There are two helper methods for initialization of the convolution layer and the max pooling layer. The Conv2d method is used from the Tensorflow API for convolution operation. ReLU is applied to the output of the convolution layer. The convolution filters are represented in four dimensions: height, width, number of channels (depth) and number of filters (filter out). There is also another variable that defines how much a filter should move, the parameter is conv filter stride. There is also a bias variable. The prediction method creates part of the computational graph that is required for making prediction via CNN. The network consists of three convolution layers, three max pooling layers, a fully connected layer, a dropout layer and an output layer. The optimize method creates part of the computation graph that is required for optimizing CNN.

3.4.3. Predictor

The predictor class is used to run a deep learning model with data. Data interface object reads the data from a raw corpus and provides data in specific number of batches. The predictor class takes the previously created data interface object as an input. The selected deep learning model is imported by the predictor class and the input data is obtained by the data interface class in batches. The predictor object runs the model on the training set, development set and test set then, prints the evaluation metrics to an output file.

3.4.4. Drivers

There are two driver scripts: “cv run” and “grid search”. Grid search is implemented by “grid search” script. In order to compare different algorithms or features, 5x2 cross validation paired t test is used and it is implemented by “cv run”.

3.4.4.1. 5x2 Cross Validation Paired t test. In order to compare two different methods, 5x2 cross validation paired t test is used and the test is implemented in “cv run” script. The script creates a data interface object and a predictor object. For two different models to be compared two predictor objects are created. Results of each run are stored and then paired t test is applied to the results.

3.4.4.2. Grid Search. Hyperparameter selection is done with grid search algorithm. Selected parameter space is stored in a text file title “grid search space.txt”. The script reads the parameter space from the file and runs the selected model for every combination in the parameter space. The results of each script are printed into an output text file.

4. EXPERIMENTS AND RESULTS

The results presented in this section focus on two areas. The first is the comparison of two neural networks widely used in extracting biomedical relations, and the second is the assessment of the effectiveness of various linguistic features. The neural networks compared in this study are CNN and BiLSTM, and the linguistic features that have been assessed are position embeddings, POS tag embeddings, IOB chunk tag embeddings and word embeddings. A grid search algorithm is used to tune the hyperparameters of the model and the effectiveness of the features is determined via a 5x2 cross validation paired t-test. The computing environment used for all of the experiments in the study is shown in Table 4.1.

Table 4.1. Computing environment.

Parameter	**Values**
Operating System	Ubuntu 18.04.2 LTS
IDE	PyCharm 2019 1.0
CPU Model	Intel 7700K 4.20GHZ
GPU Model	GTX1060
GPU Memory	6GB
Python Version	3.6.1
CUDA Version	10
Tensorflow Version	1.13.1

4.1. Training Domain of Word Embeddings

Word2vec is a widely used method for word representation in Natural Language Processing [42]. The focus of the experiments in this section is to assess the significance of the training set domain to word embedding effectiveness when the embeddings are used in the biomedical relation extraction task. The effect of the training set domain

in embedding impact is analyzed with two different neural network models: BiLSTM and CNN. Two different domains are compared in the experiments: Wikipedia and biomedical articles. Word embeddings that are trained from biomedical articles are obtained from the work of Chiu *et al.* [40]. Network parameters that are used in the work of Chie *et al.* are depicted in Table 4.2. The same parameters are used in the training of word embeddings with Wikipedia articles. Gensim library is used to train the word embeddings from Wikipedia articles [43]. Negative sampling is selecting a subset of the negative output nodes to be updated randomly. The vector dimension defines the number of nodes in the hidden layer. The learning rate is a scalar that defines the effect of the gradient to weight update. Sub-sampling is filtering frequent words randomly. Context window size refers to the window size in the creation of word pairs in training. Minimum-count defines the minimum number of occurrences for a word to be included in the dataset.

Table 4.2. Word2vec network parameters.

Parameter	**Values**
Architecture	skip-gram
Negative sample size (neg)	10
Vector dimension (dim)	200
Learning Rate (alpha)	0.05
Sub-sampling (samp)	1e-4
Context window size (win)	2
Minimum-count(min-count)	5

4.1.1. Bidirection LSTM

Two different domains are compared via the BiLSTM model that is explained in Section 3.3.1. Hyperparameter values of the BiLSTM network are depicted in Table 4.4. 5x2 cross validation t-test is used to determine if the difference is significant or not. The total dataset is shuffled and divided into two parts in each iteration.

Table 4.3. Results of word embedding comparison with BiLSTM.

Iteration	**Fold**	**F1-Measure**	
		Wikipedia	**Pubmed**
1	1	0.654	0.686
1	2	0.646	0.693
2	1	0.630	0.671
2	2	0.542	0.653
3	1	0.662	0.691
3	2	0.643	0.692
4	1	0.644	0.683
4	2	0.651	0.688
5	1	0.635	0.660
5	2	0.640	0.635
Mean		0.637	0.681
Standard Deviation		0.037	0.014
t value		4.8645	

Table 4.4. BiLSTM model parameters used in word embedding comparison.

Network Parameter	**Value**
Batch Size	50
LSTM Hidden Unit Size	128
Word Embedding Size	200
Learning Rate	0.001

The f1-measure value of each run is listed in Table 4.3. The t value is 4.8645 and the confidence interval is selected as 95%. The difference is statistically significant thus, we can conclude that using a biomedical domain for word embedding training is beneficial if the selected model is BiLSTM.

4.1.2. Convolutional Neural Network

Different domains for word embedding training are compared in Section 4.1.1 with BiLSTM. The CNN model that is explained in Section 3.3.2 is used in this section to compare different domains for word2vec training.

CNN model parameters are shown in Table 4.5. 5x2 cross validation t-test results are shown in Table 4.6. F1-measure values are lower than the values obtained by the BiLSTM values. The t value is 4.2181 and a 95% confidence interval is selected. The difference between domains is statistically significant and we can conclude that using a biomedical domain for word embedding training is beneficial with the CNN model similar to the BiLSTM model.

Table 4.5. CNN model parameters used in word embedding comparison.

Network Parameter	**Value**
Batch Size	50
Filter 1 Size	2, 200
Filter 2 Size	3, 200
Filter 3 Size	4, 200
Word Embedding Size	200
Learning Rate	0.001
Hidden Unit Size	2048

Table 4.6. Results of word embedding comparison with CNN.

Iteration	**Fold**	**F1-Measure**	
		Wikipedia	**Pubmed**
1	1	0.528	0.541
1	2	0.504	0.531
2	1	0.529	0.541
2	2	0.512	0.539
3	1	0.519	0.533
3	2	0.537	0.537
4	1	0.514	0.517
4	2	0.516	0.547
5	1	0.478	0.510
5	2	0.509	0.545
Mean		0.517	0.534
Standard Deviation		0.018	0.012
t value		4.2181	

4.2. Trainable Word Embeddings

Pre-trained word embeddings are used in all experiments. A neural network updates the network weights through an optimization algorithm. However, the model can continue to learn word embeddings. The effect of word embedding training is analyzed with both of the models. Only word embeddings are used as an input feature in this set of experiments.

4.2.1. Bidirectional LSTM

The effect of training word embeddings through model optimization is analyzed with the BiLSTM model in this section. The hyperparameter values of the BiLSTM model are shown in Table 4.8. 5x2 cross validation f1-measure results are shown in Table 4.7. The confidence interval is selected as 95% and the t value is 4.5480 thus, the difference between using trainable and fixed word embeddings is statistically significant based on the paired t-test. We can conclude that training word embedding through network optimization is beneficial when using BiLSTM.

Table 4.7. Results of trainable vs. fixed word embedding comparison with BiLSTM.

Iteration	**Fold**	**F1-Measure**	
		Trainable Word Embedding	**Fixed Word Embedding**
1	1	0.779	0.675
1	2	0.782	0.682
2	1	0.776	0.688
2	2	0.773	0.680
3	1	0.775	0.632
3	2	0.785	0.678
4	1	0.664	0.556
4	2	0.668	0.547
5	1	0.695	0.699
5	2	0.687	0.675
Mean		0.739	0.662
Standard Deviation		0.052	0.068
t value		4.5480	

Table 4.8. BiLSTM parameters in trainable vs. fixed word embedding comparison.

Network Parameter	**Value**
Batch Size	50
LSTM Hidden Unit Size	128
Word Embedding Size	200
Learning Rate	0.001

4.2.2. Convolutional Neural Network

The effect of trainable word embeddings is also analyzed with convolutional neural networks. CNN parameters that are used in 5x2 cross validation are listed in Table 4.9 and the result of each run are shown in Table 4.10. The confidance interval is selected as 95% and t value is 6.3202. The difference is significant based on paired t-test.

Table 4.9. CNN parameters in trainable vs. fixed word embedding comparison.

Network Parameter	**Value**
Batch Size	50
Filter 1 Size	2, 200
Filter 2 Size	3, 200
Filter 3 Size	4, 200
Word Embedding Size	200
Learning Rate	0.001
Hidden Unit Size	2048

Table 4.10. Results of trainable vs. fixed word embedding comparison with CNN.

Iteration	**Fold**	**F1-Measure**	
		Trainable Word Embedding	**Fixed Word Embedding**
1	1	0.559	0.480
1	2	0.552	0.467
2	1	0.554	0.480
2	2	0.559	0.485
3	1	0.539	0.503
3	2	0.485	0.458
4	1	0.537	0.491
4	2	0.590	0.523
5	1	0.427	0.424
5	2	0.536	0.492
Mean		0.534	0.480
Standard Deviation		0.045	0.026
t value		6.3202	

4.3. Position Embedding Experiments

The position embedding of a token refers to its distance to the first and second entity. The experiments in this section aim to determine the importance of position embeddings in biomedical relation extraction. Position embeddings are derived from raw data. The importance of position embeddings is tested with BiLSTM and CNN. The grid search algorithm is applied to find the best set of hyperparameters.

4.3.1. Bidirectional LSTM

The effect of the position feature is firstly evaluated with BiLSTM. The grid search algorithm is used to find the optimal hyperparameter set of the model. The grid search space is shown in Table 4.12. Some results are selected for reporting and the selected results are listed in Table 4.11. The training and development sets are used in grid search, the model is trained with the training set and the models are compared on the development set. 5x2 cross validation paired t-test is used to determine the effect of the position feature. The training, development and test sets are concatenated and spitted into two. Each fold is used as a test set. 5x2 cross validation results are shown in Table 4.13. The confidance interval is selected as 95% and the t value is 7.1616 which shows that the difference is extremely significant. We are able to conclude that position features are beneficial with BiLSTM in the biomedical relation extraction task.

Table 4.11. Selected BiLSTM grid search results for position embedding.

Position Embedding Size	**Learning Rate**	**Number of Units**	**Precision**	**Recall**	**F1-Measure**
10	0.001	128	0.62	0.61	0.62
20	0.001	128	0.61	0.61	0.61
10	0.001	256	0.61	0.63	0.62
20	0.001	512	0.60	0.64	0.62

Table 4.12. BiLSTM grid search space for position embedding experiments.

Parameter	**Values**
Hidden Unit Size	128, 256, 512
Position Embedding Size	10, 20, 50, 100
Learning Rate	0.01, 0.001

Table 4.13. Results of position embedding experiments with BiLSTM.

Iteration	**Fold**	**F1-Measure**	
		Position Embedding	**No Position Embeddings**
1	1	0.789	0.772
1	2	0.793	0.777
2	1	0.789	0.783
2	2	0.789	0.777
3	1	0.776	0.746
3	2	0.779	0.765
4	1	0.783	0.775
4	2	0.784	0.766
5	1	0.793	0.777
5	2	0.788	0.775
Mean		0.786	0.771
Standard Deviation		0.005	0.010
t value		7.1616	

4.3.2. Convolutional Neural Network

The BiLSTM model can benefit from position information of the token. The position feature is also tested with CNN in this section. The grid search algorithm is used to find the best set of hyperparameters. Only hidden unit, filter size, and position embedding size are selected to be included in the grid search, all hyperparameters could not be included in the grid search because of computational limitations. The hidden unit size represents the number of hidden units in the fully connected layer. Filter size represents the number of each filter type in the convolution layer. The grid search space is depicted in Table 4.15. The selected grid search results are shown in Table 4.14. The

confidence interval is selected as 95% and the t value is 0.9525 thus, the difference is not statistically significant. We concluded that the position feature of a token is not beneficial with CNN contradictory to the results obtained with the BiLSTM model.

Table 4.14. Selected CNN grid search results for position embedding.

Position Embedding Size	**Number of Units**	**Number of Filters**	**Precision**	**Recall**	**F1-Measure**
10	2048	200	0.48	0.69	0.56
10	2048	250	0.50	0.65	0.57
50	2048	250	0.50	0.58	0.54
20	2048	300	0.49	0.64	0.56
50	2048	300	0.49	0.64	0.56
10	2048	350	0.49	0.69	0.57
20	4096	150	0.54	0.54	0.54
10	4096	200	0.49	0.59	0.53
10	4096	250	0.51	0.62	0.56
10	4096	300	0.50	0.65	0.57
50	4096	300	0.45	0.75	0.57

Table 4.15. CNN grid search space for position embedding experiments.

Parameter	**Values**
Hidden Unit	2048, 4096
Filter Size	150, 200, 250, 300, 350
Position Embedding Size	10, 20, 50, 100

Table 4.16. Results of position embedding experiments with CNN.

Iteration	**Fold**	**F1-Measure**	
		Position Embedding	**No Position Embeddings**
1	1	0.660	0.668
1	2	0.662	0.659
2	1	0.647	0.668
2	2	0.675	0.703
3	1	0.685	0.665
3	2	0.681	0.655
4	1	0.653	0.690
4	2	0.650	0.664
5	1	0.651	0.655
5	2	0.662	0.658
Mean		0.663	0.669
Standard Deviation		0.013	0.015
t value		0.9525	

4.4. POS Tag Embedding Experiments

Part of speech tag is a linguistic feature of a token and the feature is obtain via NLTK. Part of speech tag is explained in Section 3.2 in detail. Two neural models are used to determine the effect of the POS tag feature in biomedical relation extraction. Grid search algorithm is used to find the best set of hyperparameters for each model.

4.4.1. Bidirectional LSTM

Part of speech tags are included into the input representation and the BiLSTM model is run with the newly created input representation. Grid search space is depicted in Table 4.19. The selected experiments from grid search are shown in Table 4.18. 5x2 cross validation is applied to determine the effect of POS tag embeddings with the BiLSTM model. The results of 5x2 cross validation are shown in Table 4.17. The confidence interval is 95% and t value is 1.6722 thus, we conclude that POS tag embeddings are not beneficial when they are used with the BiLSTM model.

Table 4.17. Results of POS tag embedding experiments with BiLSTM.

Iteration	**Fold**	**F1-Measure**	
		POS Tag Embedding	**No POS Tag Embeddings**
1	1	0.787	0.787
1	2	0.794	0.793
2	1	0.795	0.794
2	2	0.792	0.797
3	1	0.790	0.790
3	2	0.799	0.791
4	1	0.788	0.792
4	2	0.714	0.697
5	1	0.726	0.721
5	2	0.721	0.703
Mean		0.769	0.766
Standard Deviation		0.038	0.041
t value		1.6722	

Table 4.18. Selected BiLSTM grid search results for POS tag embedding.

Position Embedding Size	**POS Tag Size**	**Number of Units**	**Precision**	**Recall**	**F1-Measure**
10	20	256	0.58	0.69	0.63
10	10	256	0.60	0.67	0.63
10	50	256	0.59	0.67	0.63
20	10	256	0.58	0.70	0.64
20	50	256	0.59	0.69	0.64

Table 4.19. BiLSTM grid search space for POS tag embedding experiments.

Parameter	**Values**
Hidden Unit Size	128, 256, 512
Position Embedding Size	10, 20, 50, 100
POS Tag Embedding Size	10, 20, 50, 100

4.4.2. Convolutional Neural Network

POS tag information of a token is not beneficial when BiLSTM model is used as a predictor as analyzed in Section 4.4.1. POS tags are also analyzed with the CNN model. Grid search is applied with CNN in order to obtain the best set of hyperparameters. The parameter space for grid search is shown in Table 4.20. Selected runs from grid search are listed in Table 4.22. 5x2 cross validation paired t-test results are shown in Table 4.21. The confidence interval is selected as 95% and the t value is 3.6809. Based on the 5x2 cross validation paired t-test, we concluded that POS tag information of a token is beneficial when CNN model is used as a predictor.

Table 4.20. CNN grid search space for POS tag embedding experiments.

Parameter	**Values**
Number of Filters	150, 200, 250, 300
Position Embedding Size	10, 20
POS Tag Embedding Size	10, 20, 50

Table 4.21. Results of POS tag embedding experiments with CNN.

Iteration	**Fold**	**F1-Measure**	
		POS Tag Embedding	**No POS Tag Embeddings**
1	1	0.612	0.588
1	2	0.625	0.609
2	1	0.619	0.603
2	2	0.618	0.606
3	1	0.609	0.607
3	2	0.602	0.576
4	1	0.593	0.565
4	2	0.589	0.588
5	1	0.601	0.604
5	2	0.607	0.601
Mean		0.608	0.595
Standard Deviation		0.011	0.015
t value		3.6809	

Table 4.22. Selected results of the CNN grid search for POS tag embedding.

Position Tag Size	POS Tag Size	Filter Number	Unit Number	Precision	Recall	F1-Measure
10	50	200	4096	0.39	0.82	0.53
10	50	250	4096	0.49	0.68	0.57
10	50	300	4096	0.62	0.31	0.41
20	50	250	4096	0.51	0.55	0.53
20	70	250	4096	0.49	0.64	0.56
10	70	250	4096	0.37	0.86	0.52
10	50	300	4096	0.62	0.31	0.41
10	50	300	8192	0.43	0.76	0.55
10	50	250	8192	0.76	0.18	0.29
10	50	250	8192	0.39	0.75	0.52

4.5. IOB Chunk Embedding Experiments

Chunking is a linguistic feature like POS tagging. Chunking refers to token grouping in the text and inside-out-beginning format is used to represent chunks. Noun phrase, verb phrase, and prepositional phrase are used in chucking. Extraction of IOB chunks and adding IOB chunk to input are explained in Section 3.2 in detail. The effect of chucking is analyzed with CNN and BiLSTM in this section.

4.5.1. Bidirectional LSTM

IOB chunk feature is included in the input representation for this set of experiments. Grid search is applied in order to find the best set of hyperparameters. The grid search space is shown in Table 4.24. The grid search results are listed in Table 4.25. 5x2 cross validation paired t-test is used to determine the effect of chunking.

The results of 5x2 cross validation are listed in Table 4.23. The confidence interval is selected as 95% and the t value is 0.2376. We concluded that the chunking is not beneficial when it is used with BiLSTM.

Table 4.23. Results of IOB tag embedding experiments with BiLSTM.

Iteration	**Fold**	**F1-Measure**	
		IOB Tag Embedding	**No IOB Tag Embeddings**
1	1	0.710	0.708
1	2	0.720	0.718
2	1	0.719	0.716
2	2	0.715	0.714
3	1	0.711	0.714
3	2	0.709	0.716
4	1	0.696	0.682
4	2	0.712	0.713
5	1	0.687	0.695
5	2	0.701	0.709
Mean		0.708	0.708
Standard Deviation		0.010	0.011
t value		0.2376	

Table 4.24. BiLSTM grid search space for IOB tag embedding experiments.

Parameter	**Values**
Number of Hidden Units	250
Position Embedding Size	10, 20
POS Tag Embedding Size	20, 50, 70
IOB Chunk Tag Embedding Size	20, 50, 70

Table 4.25. Selected BiLSTM grid search results for IOB embedding.

Position Size	POS Tag Size	IOB Tag Size	Precision	Recall	F1-Measure
10	20	20	0.59	0.67	0.63
20	70	50	0.60	0.65	0.63
20	50	20	0.62	0.60	0.61
10	70	50	0.63	0.58	0.60

4.5.2. Convolutional Neural Network

Chunking is not beneficial when it is used with BiLSTM as explained in Section 4.5.1. Chunking is also tested with convolutional neural network, in order to determine the effect of chunking on biomedical relation extraction.

Table 4.26. Selected CNN grid search results for IOB embedding.

Filter Number	Position Tag Size	POS Tag Size	IOB Tag Size	Precision	Recall	F1-Measure
150	10	10	10	0.46	0.69	0.55
150	20	20	10	0.45	0.69	0.55
200	10	20	50	0.34	0.92	0.50
200	10	50	50	0.40	0.79	0.53
200	20	50	20	0.49	0.65	0.56
250	20	50	10	0.47	0.63	0.54
300	20	50	20	0.44	0.72	0.55
300	50	50	50	0.40	0.84	0.54

Table 4.27. CNN grid search space for IOB tag embedding experiments.

Parameter	**Values**
Number of Filters	150, 200, 250, 300
Position Embedding Size	10, 20
POS Tag Embedding Size	10, 20, 50
IOB Chunk Tag Embedding Size	10, 20, 50

Table 4.28. Results of IOB tag embedding experiments with CNN.

Iteration	**Fold**	**F1-Measure**	
		IOB Tag Embedding	**No IOB Tag Embeddings**
1	1	0.630	0.624
1	2	0.612	0.602
2	1	0.627	0.638
2	2	0.611	0.621
3	1	0.618	0.630
3	2	0.622	0.604
4	1	0.602	0.615
4	2	0.602	0.561
5	1	0.606	0.610
5	2	0.595	0.600
Mean		0.612	0.610
Standard Deviation		0.011	0.021
t value		0.3668	

Noun phrase, verb phrase and prepositional phrase are included in the input representation. POS tag embeddings and position embeddings are also included in the input. Grid search is applied to find the optimal solution and grid search space is shown in Table 4.27. Selected experiments from grid search are shown in Table 4.26.

The best recall score based on the grid search results is obtained by convolutional neural network with IOB chunk tag embeddings. The results of 5x2 cross validation paired t-test are shown in Table 4.28. Confidence interval is selected as 95% and the t value is 0.3668. The difference is not statistically significant thus we concluded that IOB chunking is not beneficial for CNN model similarly BiLSTM.

4.6. Test Set Evaluations

The grid search applied through Section 4.1 to Section 4.5 used the training and development sets. For 5x2 cross validation paired t-test, the whole dataset is concatenated and splitted into two parts. In this section, the training set is used for training and the test set is only used for testing purposes. The best set of hyperparameters is selected for both of the models. The best set of hyperparameters is obtained by grid search that is evaluated in Section 4.1 to Section 4.5. The model parameters of CNN are shown in Table 4.32 and the model parameters of BiLSTM are shown in Table 4.31.

Subsampling is also tested in this section. Negative labelled candidate relations that occur in positively labelled candidate relations are filtered for subsampling. Subsampling results are listed in Table 4.29. We concluded that subsampling method is not beneficial when it is tested on the test set. Both of the models perform similarly when they are both trained with training set and evaluated on the test set. Average precision is also tested in this section. The results for average precision are listed in Table 4.30.

Table 4.29. Evaluation of selected best models on the test set.

Model	**Sub-sampling**	**Precision**	**Recall**	**F1-measure**
CNN	Yes	0.46	0.69	0.55
CNN	No	0.51	0.66	0.57
BiLSTM	Yes	0.41	0.88	0.56
BiLSTM	No	0.55	0.57	0.56

Table 4.30. Average precision of selected models on the test set.

Model	**Sub-sampling**	**0.9**	**0.8**	**0.7**	**0.6**	**0.5**
CNN	Yes	0.46	0.46	0.46	0.46	0.46
CNN	No	0.56	0.55	0.55	0.55	0.55
BiLSTM	Yes	0.56	0.53	0.51	0.49	0.48
BiLSTM	No	0.65	0.64	0.63	0.62	0.61

Table 4.31. The selected best BiLSTM model.

Parameter	**Value**
Hidden unit size	256
Position embedding size	10
Batch size	50
Learning rate	0.001

Table 4.32. The selected best CNN model.

Parameter	**Value**
Filter out	250
Hidden unit size	4096
Filter1 size	(2, 260)
Filter2 size	(3, 260)
Filter3 size	(4, 260)
Position embedding size	10
POS tag embedding size	50
Batch size	50
Learning rate	0.001

5. CONCLUSION

This thesis aimed to assess the effect of linguistic features and propose an artificial neural network model for the task of biomedical relation extraction. The experiments in Section 4.1 and 4.2 focus on improving the effectiveness of word embeddings for biomedical relation extraction. The experiments in Sections 4.3, 4.4, and 4.5 are conducted in order to assess the effects of including POS tag, IOB chunking, and position information in the input representation. In light of the experiments that are conducted in this thesis, valuable information about different neural network models and input features can be inferred. With regard to neural network models, BiLSTM is found to be superior to the CNN model when evaluated on the development set. However, the models perform similarly when they are evaluated on the test set. Choosing the biomedical domain for word embedding training has been shown to improve test performance. Training word embeddings through model optimization has been shown to improve performance for both models.

The word embedding experiments are aimed to compare word embeddings that are trained with different training sets. Biomedical word embeddings are obtained from the work of Chiu *et al.* [40] and Wikipedia word embeddings are trained with the same parameters as the biomedical word embeddings. They are compared with both of the models. The analysis of the effect of the training domain to word embedding effectiveness for biomedical relation extraction is done in Section 4.1. We concluded that the biomedical word embeddings are significantly improving the performance of both models in biomedical relation extraction. Context information that is obtained from the biomedical articles could be the reason behind the increase in the accuracy.

Section 4.2 aims to assess the effect of training word embeddings through the model. Pre-trained word embeddings are used in this work. However, word embeddings can also be trained through model optimization. We concluded that adding word embeddings as variables to be optimized significantly increases the performance.

Position embeddings are valuable information in relation extraction tasks. The preparation of position embeddings is explained in section 3.2 in detail. Position embeddings are added to word embeddings in input representation and tested with two different models. Section 4.3 contains the analysis of the effect of position embedding in biomedical relation extraction. Grid search is applied to each model in order to obtain the optimal parameters. We concluded that the position information of a token is beneficial for the BiLSTM model. However, we realized that CNN can not benefit from position information as well as BiLSTM. 5x2 cross validation paired t-test is used to determine the effect of position information. The CNN model has filters in three different sizes, where each filter learns relations merely between 2, 3 or 4 tokens. Thus the idea of the CNN model is based on localisation, which may be the reason why it benefits less from the position information compared to BiLSTM.

POS tags are type of linguistic information derived from raw text. POS tags are explained in Section 3.2. POS tag analysis is done in Section 4.4. Grid search is used to find the optimal hyperparameter set. POS tags are added to input representation and the models are evaluated with the new representation. 5x2 cross validation is used to determine the effect of POS tag embeddings. Our results showed that BiLSTM can not benefit from POS tag embeddings but POS tag information of a token is beneficial for CNN. The reason why the BiLSTM model does not benefit from POS information could be that BiLSTM learns this information internally.

IOB chunks are also linguistic information like POS tags. IOB chunks are explained in section 3.2 in detail. 5x2 cross validation paired t-test is used to assess the effect of IOB chunk tags. Grid search is applied to find the optimal set of hyperparameters for both of the models. The analysis is done in Section 4.5. We observe that IOB chunks are not effective for both of the models. Chunking information could have been learned by both of the models internally from the POS information. This might be the reason why providing chunking information to the models externally did not result in increase in the F-measure performance in Section 4.5.

Section 4.6 contains the performance of the selected best models over the test set. We observed that both of the models perform similarly to each other with the best set of hyperparameters. The subsampling method is also evaluated in Section 4.6. We observed that subsampling method is not effective.

As future work, we plan to investigate the use of information that is derived from dependency parsing. Also, new layers can be added to the deep learning models and new models can be added to the comparison in this thesis. We concluded that BiLSTM and CNN perform similarly to each other on the test set. However, BiLSTM requires none of the linguistic information that are expensive to obtain in terms of time. BiLSTM with merely word embeddings and position information performs similarly to CNN with word embeddings, position information, POS tag information, and IOB tag information.

REFERENCES

1. U.S. National Library of Medicine, *Citations Added to MEDLINE® by Fiscal Year*, 2019, `https://www.nlm.nih.gov/bsd/stats/cit_added.html`, accessed in June 2019.

2. Gilson, M. K., T. Liu, M. Baitaluk, G. Nicola, L. Hwang and J. Chong, "BindingDB in 2015: A public database for medicinal chemistry, computational chemistry and systems pharmacology", *Nucleic Acids Research*, Vol. 44, No. D1, pp. D1045–D1053, 2016.

3. Li, Y., Z. Liu, J. Li, L. Han, J. Liu, Z. Zhao and R. Wang, "Comparative assessment of scoring functions on an updated benchmark: 1. Compilation of the test set", *Journal of chemical information and modeling*, Vol. 54, No. 6, pp. 1700–1716, 2014.

4. Hu, L., M. L. Benson, R. D. Smith, M. G. Lerner and H. A. Carlson, "Binding MOAD (Mother Of All Databases)", *Proteins: Structure, Function, and Bioinformatics*, Vol. 60, No. 3, pp. 333–340.

5. Ahmed, A., R. D. Smith, J. J. Clark, J. B. Dunbar, Jr and H. A. Carlson, "Recent improvements to Binding MOAD: a resource for protein–ligand binding affinities and structures", *Nucleic Acids Research*, Vol. 43, No. D1, pp. D465–D469, 2015.

6. Blundell, T. L. and K. Mizuguchi, "Structural genomics: an overview", *Progress in biophysics and molecular biology*, Vol. 5, No. 73, pp. 289–295, 2000.

7. Jeremy M Berg, J. L. T. and L. Stryer, *Biochemistry*, W.H. Freeman, New York, 5th edn., 2002.

8. Gilson, M. K. and H.-X. Zhou, "Calculation of protein-ligand binding affinities", *Annu. Rev. Biophys. Biomol. Struct.*, Vol. 36, pp. 21–42, 2007.

9. Trott, O. and A. J. Olson, "AutoDock Vina: Improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading", *J. Comput. Chem*, pp. 455–461.

10. Li, Y., Z. Liu and R. Wang, "Test MM-PB/SA on true conformational ensembles of protein−ligand Complexes", *Journal of Chemical Information and Modeling*, Vol. 50, No. 9, pp. 1682–1692, 2010.

11. Hsin, K.-Y., S. Ghosh and H. Kitano, "Combining machine learning systems and multiple docking simulation packages to improve docking prediction reliability for network pharmacology", *PloS one*, Vol. 8, No. 12, p. e83922, 2013.

12. Rosenblatt, F., "The perceptron: a probabilistic model for information storage and organization in the brain.", *Psychological review*, Vol. 65, No. 6, p. 386, 1958.

13. Minsky, M. and S. Papert, *Perceptrons: An Introduction to Computational Geometry*, MIT Press, Cambridge, MA, USA, 1969.

14. Rumelhart, D. E., G. E. Hinton and R. J. Williams, "Learning representations by back-propagating errors", *Nature*, Vol. 323, No. 6088, pp. 533–536, oct 1986.

15. Hochreiter, S., "Untersuchungen zu dynamischen neuronalen Netzen", *Diploma, Technische Universität München*, Vol. 91, No. 1, 1991.

16. Hochreiter, S. and J. Schmidhuber, "Long short-term memory", *Neural computation*, Vol. 9, No. 8, pp. 1735–1780, 1997.

17. Gers, F. A., J. Schmidhuber and F. A. Cummins, "Learning to Forget: Continual Prediction with LSTM", *Neural Computation*, Vol. 12, pp. 2451–2471, 2000.

18. Abadi, M., A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems", *arXiv preprint arXiv:1603.04467*, 2016.

19. Zaremba, W., I. Sutskever and O. Vinyals, "Recurrent neural network regularization", *arXiv preprint arXiv:1409.2329*, 2014.

20. Sak, H., A. Senior and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling", *Fifteenth annual conference of the international speech communication association*, 2014.

21. Lecun, Y., L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition", *Proceedings of the IEEE*, Vol. 86, No. 11, pp. 2278–2324, Nov 1998.

22. Matos, S., "Extracting chemical–protein interactions using long short-term memory networks", *Proceedings of the BioCreative VI Workshop*, pp. 151–154, 2017.

23. Liu, S., F. Shen, Y. Wang, M. Rastegar-Mojarad, R. K. Elayavilli, V. Chaudhary and H. Liu, "Attention-based neural networks for chemical protein relation extraction", *Training*, Vol. 1020, No. 25.247, p. 4157, 2017.

24. Warikoo, N., Y. Chang, P. Lai *et al.*, "CTCPI–convolution tree kernel based chemical-protein interaction detection", *Proceedings of 2017 BioCreative VI Workshop, Maryland, USA, October 2017*, pp. 168–171, 2017.

25. Yüksel, A., H. Öztürk, E. Ozkirimli, A. Özgür *et al.*, "CNN based chemical-protein interactions classification", *Proceedings of the BioCreative VI Workshop, Bethesda, MD*, pp. 184–186, 2017.

26. Corbett, P. and J. Boyle, "Improving the learning of chemical-protein interactions from literature using transfer learning and specialized word embeddings", *Database*, Vol. 2018, 2018.

27. Tripodi, I., M. Boguslav, N. Hailu and L. Hunter, "Knowledge-base-enriched relation extraction", *Proceedings of the Sixth BioCreative Challenge Evaluation Workshop. Bethesda, MD USA*, Vol. 1, pp. 163–6, 2017.

28. Mehryary, F., J. Björne, T. Salakoski and F. Ginter, "Combining support vector machines and lstm networks for chemical protein relation extraction", *Proceedings of the BioCreative VI Workshop*, pp. 176–180, 2017.

29. Lim, S. and J. Kang, "Chemical–gene relation extraction using recursive neural network", *Database*, Vol. 2018, 2018.

30. Wang, W., X. Yang, Y. Xing, C. Wu and Z. Song, "Extracting chemical-protein interactions via bidirectional long short-term memory network", *Proceedings of the BioCreative VI Workshop, Bethesda, MD*, pp. 171–174, 2017.

31. Peng, Y., A. Rios, R. Kavuluru and Z. Lu, "Chemical-protein relation extraction with ensembles of SVM, CNN, and RNN models", *arXiv preprint arXiv:1802.01255*, 2018.

32. Verga, P. and A. McCallum, "Predicting chemical protein relations with biaffine relation attention networks", *Proceedings of the BioCreative VI Workshop*, 2017.

33. Lung, P.-Y., Z. He, T. Zhao, D. Yu and J. Zhang, "Extracting chemical–protein interactions from literature using sentence structure analysis and feature engineering", *Database*, Vol. 2019, 2019.

34. Krallinger, M., O. Rabal, S. A. Akhondi *et al.*, "Overview of the BioCreative VI chemical-protein interaction Track", *Proceedings of the sixth BioCreative challenge evaluation workshop*, Vol. 1, pp. 141–146, 2017.

35. Peng, Y. and Z. Lu, "Deep learning for extracting protein-protein interactions from biomedical literature", *arXiv preprint arXiv:1706.01556*, 2017.

36. Kavuluru, R., A. Rios and T. Tran, "Extracting drug-drug interactions with word and character-level recurrent neural networks", *2017 IEEE International Conference on Healthcare Informatics (ICHI)*, pp. 5–12, IEEE, 2017.

37. Peng, Y., A. Rios, R. Kavuluru and Z. Lu, "Extracting chemical–protein relations with ensembles of SVM and deep learning models", *Database*, Vol. 2018, 07 2018.

38. Mikolov, T., I. Sutskever, K. Chen, G. Corrado and J. Dean, "Distributed Representations of Words and Phrases and Their Compositionality", *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pp. 3111–3119, Curran Associates Inc., USA, 2013.

39. Bird, S., E. Klein and E. Loper, *Natural Language Processing with Python*, O'Reilly Media, Inc., 1st edn., 2009.

40. Chiu, B., G. Crichton, A. Korhonen and S. Pyysalo, "How to train good word embeddings for biomedical NLP", *Proceedings of the 15th workshop on biomedical natural language processing*, pp. 166–174, 2016.

41. Kim, Y., "Convolutional neural networks for sentence classification", *arXiv preprint arXiv:1408.5882*, 2014.

42. Mikolov, T., I. Sutskever, K. Chen, G. S. Corrado and J. Dean, "Distributed representations of words and phrases and their compositionality", C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani and K. Q. Weinberger (Editors), *Advances in Neural Information Processing Systems 26*, pp. 3111–3119, Curran Associates, Inc., 2013.

43. Řehůřek, R. and P. Sojka, "Software Framework for Topic Modelling with Large Corpora", *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pp. 45–50, ELRA, Valletta, Malta, May 2010.