Indian Statistical Institute
Kolkata

M.Tech. (Computer Science) Dissertation

# Recurrent Neural Network Based Information Extraction for Cancer Genetics

A dissertation submitted in partial fulfilment of the requirements
for the award of Master of Technology
in
Computer Science

*Author:*
Arkadeep Banerjee
Roll No: CS 1327

*Mentor:*
Dr. Utpal Garain
Computer Vision and
Pattern Recognition Unit

## CERTIFICATE

**Student : Arkadeep Banerjee (CS1327)**
**Topic : Recurrent Neural Network based Information Extraction for Cancer Genetics**
**Mentor : Dr. Utpal Garain**

This is to certify that the thesis titled ***Recurrent Neural Network based Information Extraction for Cancer Genetics*** submitted by Arkadeep Banerjee in partial fulfilment for the award of the degree of Master of Technology is a bonafide record of work carried out by him under my supervision. The thesis has fulfilled all the requirements as per the regulations of this Institute and, in our opinion, has reached the standard needed for submission. The results embodied in this thesis have not been submitted to any other university for the award of any degree or diploma.

**Dr. Utpal Garain**

**Date :** 13th *July*, 2015

# ABSTRACT

With the burgeoning load of scholarly articles related to Cancer Genetics being made available every year, its the need of the hour that a robust system be developed in order to extract information from these article so that the actual want of the user, expressed via his/her query, can be processed accordingly and the articles presented have a high *relevance* to the user. While being similar in nature to other Information Extraction task in the BioNLP domain, the CG extraction task has to be generalized across *events* capturing interactions between *entities* across the entire biological hierarchy from simple chemical to organism. This paper is invested in exploring the design and implementation of a supervised learning based sequence classification technique to advance the automatic extraction of information(*events and arguments*) from statements on the biological processes. The paper discusses the use of efficient word embeddings in vector space via distributed representation [Mikolov et al., 2013a] [Mikolov et al., 2013b] on the *pre-processed* CG corpus and derive semantic relation between words, which is later fed as input to *Recurrent Neural Networks* (RNNs) and *Long Short Term Memory Networks* (LSTMN) [Hochreiter and Schmidhuber, 1997] for extracting information (viz. events and its arguments), and the results are compared to the current state-of-the-art techniques.

# Acknowledgements

First and foremost, I am grateful to Dr. Utpal Garain, my mentor for not only taking me under his tutelage and finding me able to work in the field of Cancer Genetics, and for providing me with encouragement and help for any hardships faced during the execution of the work, but also for transcending beyond the role of just a mentor, and becoming a friend, philosopher and guide, when I needed it most.

I am ever-indebted to my father, Late Somenath Banerjee, who had motivated me, still motivates me and will continue to motivate me when I need it most and show me hope in times of despair, despite being physcially absent from this material world. I am grateful for my mother, Nandita Banerjee and sister, Asmita Banerjee, for taking care of me throughout the journey and providing me with much needed mental support, as I often stayed up and awake, well beyond the Devils' Time in pursuit of this discourse. I am also thankful for my uncle, Tapas Poddar, for providing me with those bursts of energy for walking that extra mile, at times when I found myself tired and drudging along, and helping me reach countless stations after station in life. I am also thankful to my aunt, Sujata Maity and my closest friend, Poulomi Maity, for pitching in with their enthusiasm whenever my pitcher ran empty, and always egging me on to be better than my yesterday's self.

Last but not least, I am grateful to Abhishek da, Arjun da for making me a part of their camaraderie, in our NLP Lab and at the same time helping me with their valuable inputs towards a better result.

# List of Tables

# List of Figures

# Contents

# Chapter 1

# Introduction

## 1.1 Motivation

Information Extraction systems in biological domain on scholarly articles have existed for quite some time, and has performed fairly well in a variety of extraction task[Pyysalo et al., 2013]. The choice of such system stacks generally revolved around *Support Vector Machines*, *Rule-based System* and/or *CRF-based tools* as the classifier. *Recurrent Neural Network* (RNN)as a part of these system stack has been largely ignored, due to many reasons such as computational complexity being dependent on the dimensions of input vectors as well as architectural dimension of the system[Burges, 1998], and for the problem of *presenting words as an input to the Neural Network System.* In the previous works, where RNNs were used in NLP domain, the feature vectors were of mainly syntactical origin. Using word as it means in the corpus had been a distant dream, until the advent of distributed representation of word embeddings by [Mikolov et al., 2013b] came into the fore. They proposed an unique method known as *Skip-Gram with Negative Sampling* (SGNS) which arrived at *better* embeddings of word in the corpus efficiently than the existing methods. This enabled us to foray into the domain of Information Extraction, using RNN and word embeddings of the entire corpus.In a way, the USP of this paper is in the fact that *RNN is being used for NLP Task, by using it on feature vectors of semantic origin rather than syntactic origin as had been done previously.*
The Concepts and the Methodology behind the implementations are described, and the results benchmarked against the top performers of the Cancer Genetics Task.

## 1.2 Organization of the Thesis

**Chapter 2** describes the Cancer Genetics Task in details as given in *http://2013.bionlp-st.org/tasks/cancer-genetics*. It discusses the ever growing field of cancer genetics in terms of the volume of data generated every year, and the need of a specialized system based on NLP technique for effective retrieval of documents, in this domain , in addition to system based on conventional IR technique. It also provides the definition of *event extraction*, in the setting of biological process, and mentions the aim of the CG Task. It further provides details of entities and events as provided by the CG Task Committee, and the *events* this paper focuses on extracting.
**Chapter 3** discusses about the teams that participated in the task. The chapter also establishes the state of the art technique as of yet, and delve into its methodology.
**Chapter 4** introduces an in-depth analysis of the proposed methodology. Starting from the motivation behind the same, we proceed to the concept of word embedding via distributed representation and how it help us in achieving the aim of the task. We then discuss the method via which word embedding is arrived at, and its parameter selection. Then we move on to shallow RNN and LSTMN based training, we fine tune the parameters of the training, and compare the result with the state-of-the-art as given in the website, for the handful of events we are considering, and reflect on the performance observed.
**Chapter 5** finally states the scope of future work, and the necessary mechanism required to achieve the same.
**Bibliography** lays out the citations that have been consulted throughout the working and construction of the thesis.

# Chapter 2

# Cancer Genetics Task

## 2.1 What is Cancer Genetics?

Cancer is a genetic disorder, where due to the *over expression* and/or *under expression* of certain genes the cell growth process is disturbed. The cause of those unexpected expression may range over a lot of factors, but is basically due to mutations of DNA in the genes and the effect of which can be seen on the the cell component to the organism itself. The study of the cause and effect of such mutations in the genes leading to unabated cell proliferation, is known as Cancer Genetics.

## 2.2 Growing Domain of Cancer Genetics

Cancer genetics is now one of the fastest expanding medical specialties. The Scientific literature on Cancer Genetics is quite extensive, and is rapidly increasing in volume, both in terms of contents and topics, each passing day. Keeping in pace with the inflating information, our understanding of biological processes involved in Cancer Genetics has kept up the pace as well. We, at present times, are closer towards understanding the role of various biological entities, in regulating the effect and outcome across the entire biological organization hierarchy, than ever before.
As of 2013, a simple query on *PubMed* for *cancer* returns **2.7 million** scientific article citations, with **140,000** citations regarding *cancer* from 2011.
In order to build and maintain a comprehensive, up-to-date knowledge base on cancer genetics, automatic support for extracting information on biological process out of the literature is required to aid the conventional IR system, in bettering their precision and recall in this domain.

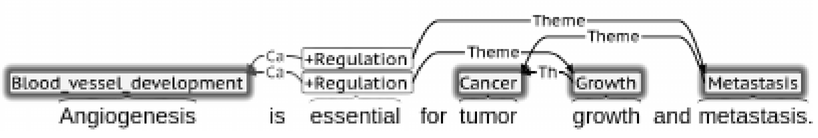


Figure 2.1: An example event with its arguments encapsulating a biological process

## 2.3 Cancer Genetics Task from BioNLP-Shared Task'13

### 2.3.1 Introduction

The CG task deals with the *extraction of events* which are relevant to cancer, all the way from simple chemical interactions between cell organelles, tissue, organs to outcomes at the level of organ system and the organism itself.
The information extraction in the Cancer Genetics manages to extend information extraction to higher levels of biological organization, which never had never had been dealt with in the previous information extraction tasks of the BioNLP Community [Nédellec et al., 2013] .
"The CG task involves the extraction of 40 event types involving 18 types of entities, defined with respect to community-standard ontologies (Pyysalo et al., 2011a; Ohta et al., 2012). The newly introduced CG task corpus, prepared as an extension of a previously introduced corpus of 250 abstracts (Pyysalo et al., 2012), consists of 600 PubMed abstracts annotated for over 17,000 events." [Nédellec et al., 2013]
The system needs to capture these biological processes, along with other details automatically from the given CG corpus. Association/ Relation between the entities also needs to be captured for our task, for e.g. Fig. 2.1 is an example from the CG website.

### 2.3.2 Definition of Events and Entities

#### 2.3.2.1 Entities

An *Entity* can be defined as a representation of a instanec belonging to any biological organization (or biological entity, for short), of various hierarchical level. Each entity posses a *type* or a class, that depicts the level of hierarchy it belongs to. For example,

- *Tumor* is an entity of type *Cancer*
- *hk-2* is an entity of type *Gene or Gene Product*
- *Liver* is an entity of type *Organ*

The types are nested according to a specified hierarchy. For example , the following types *Simple Chemical*, *Gene or Gene Product*, *Amino Acid* falls under the broader hierarchy of *Molecule*.

#### 2.3.2.2 Events

An *Event* can be defined *recursively* as interactions between *entities* and/or *events*. The interacting *entities* or *events*, are known as the *arguments* of the parent *event*. The *type* and *number* of arguments depend on the parent *event* type. Events are the encapsulation of biological processes in text.
In a textual corpus, a representation of a Event has two features:

- **Trigger Word** : In simple terms, the words denoting the *interaction* between entities-entities or entities-events or events-events are defined as trigger words, i.e. the trigger word denotes the event itself of a particular type in the document being evaluated.
*All the events and entities will be denoted in the subscripted form as* ***Event***$^{EventType}$ *and* ***Entity Word***$_{EntityType}$
In the Fig. 2.1, **Angiogenesis**$^{BloodVesselDevelopment}$, **growth**$^{Growth}$, **essential**$^{Regulation}$ and **metastasis**$^{Metastasis}$ are the trigger words for their respective event types i.e, **Angiogenesis** is the trigger word denoting the presence of an event of *Blood Vessel Development* type.

- **Arguments** : Besides capturing and classifying the *trigger words*, the system needs to capture the the *entities* and other *events* that takes part in the interaction. The *arguments* have several roles as defined by the events, of which they are a part. The main or the central argument(s) are denoted as **theme.**, with other auxiliary arguments also present at times depending on the nature of the event. As mentioned before, in the description of a single event, arguments can be entities from various levels of biological organization hierarchy, as well as other events.
In the previous example, the entity **tumor**$_{Cancer}$ is an argument of **theme** type for both the events denoted by **growth**$^{Growth}$ and **metastasis**$^{Metastasis}$. Both thesse events **growth**$^{Growth}$ and **metastasis**$^{Metastasis}$, are in themselves an argument of **theme** type of the event denoted by **essential**$^{Regulation}$, where as the event denoted by **Angiogenesis**$^{BloodVesselDevelopment}$, which *does not have any* arguments, is also an argument of **cause** type for the event denoted by **essential**$^{Regulation}$.

### 2.3.3 Requirement of NLP Techniques

A biological process consists of a lot of *biological entities* across the complete biological organization hierarchies e.g. Organism, Tissue, Cellular Component, Nucleic Acids, Simple Chemical, etc. *interacting* e.g. regulating, inhibiting, expressing, etc., in the presence of each other. A system intended to mine such information must do so across the entire biological organization hierarchy.

The requirement for such a system goes way beyond the usual search techniques as can be afforded by conventional Information Retrieval System alone, where just the occurrence of terms of the query in the document is not sufficient for it to be adjudged relevant. A specific *relation* needs to exist between the query words in the document, the nature of which lends itself for the use of many conventional *(and unconventional)* Natural Language Processing Techniques. For example, consider the following query

*growth of* ***tumor*** *in* ***blood vessel***,

chances are high that the documents returned by an unaided traditional IR system would contain more documents about *growth of* ***blood vessel*** *in* ***tumor*** than documents on *growth of* ***tumor*** *in* ***blood vessel***, simply for the sheer volume of the documents dealing with the former topic than the latter one. Therefore it leads to low recall in the top k documents, and a low precision as well.

### 2.3.4 Aim of the Task

The Event Extraction task has two parts:

- **Identifying Trigger Words** : As mentioned before, the trigger words marks the presence of an event. The first part of the task deals with identifying the trigger words, which broadly falls into a *sequence classification problem.* Besides identifying the trigger words for events, words denoting 2 entities also needs to be identified, viz dna domain or region and protein domain or region.
- **Identifying Arguments** : Following the extraction of event based on trigger words, the arguments, if any, needs to be identified. Events can have multiple types of arguments i.e, **theme, cause, atLocation**, etc. Arguments of a particular type has the following features etc :
  - *Number of Arguments* : In the previous example, **Angiogenesis**$^{BloodVesselDevelopment}$ has no arguments, although it can have at

most one arguments. As of current definition, arguments can be at most one, exactly one or multiple ones.

  – *Nature* : Arguments can be either *entity* or *events* or both, across multiple hierarchies, depending on the *type* of the current event.

For Example,

..*growth* of *tumors* in the *blood vessel*..

here *growth* is a trigger word of event type **Growth**, and its **theme** argument is *tumors*, which is an entity. Also the aforesaid process has another argument of **atLocation** type to be captured, which is *blood vessel*, which is also an entity type.
This fine tunes the document retrieval where a user might want to see articles based on *growth* of *tumors* in *blood vessel*, and **not** articles based on *growth* of *blood vessel* in *tumors*, thereby improving recall and precision at the same time.

- **Event Modification** : The last task is to determine whether a particular event is a *negation* or *speculation* type. Since much of the success at this stage depends on the accuracy of the previous stages, this paper lays emphasis on improving the recall and precision of the previous tasks only, with scope of extending the system to include Event Modification as a Future Work.

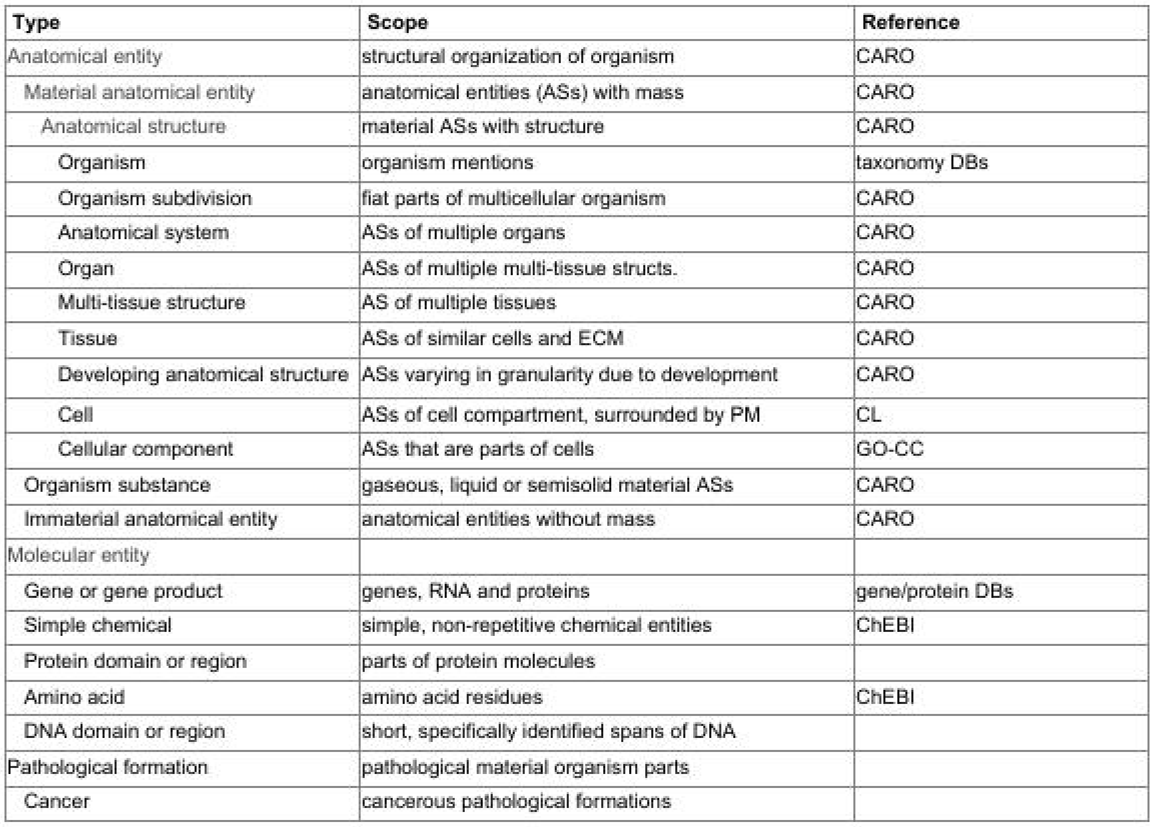
For the purpose of the Task a total of 40 events have been proposed, along with nature and number of their arguments. However for the scope of this paper, we shall be considering only 4 of them that captures as much of diversity as possible in its argument type and numbers. In the given Table 2.1, we discuss those events along with their arguments,

| **Event Name** | **Arguments** |
|---|---|
| Blood Vessel Development | *Theme?*(Ana/Path), *AtLoc?*(Ana/Path) |
| Cell Death | *Theme?*(Cell) |
| Cell Transformation | *Theme*(Cell),*AtLoc?*(Ana/Path) |
| Gene Expression | *Theme+*(Gene or Gene Product) |

Table 2.1: A list of events along with the type and number of their arguments, which is considered for this paper.

*+* denotes any number of arguments of that type, *?* denotes at most one argument of that type and *no* modifier denotes exactly one argument.

Figure 2.2: A tabular representation of the annotated Entity List provided as a part of CG Task, with their hierarchies described via greyed out entries.



| Type | Scope | Reference |
|---|---|---|
| Anatomical entity | structural organization of organism | CARO |
| Material anatomical entity | anatomical entities (ASs) with mass | CARO |
| Anatomical structure | material ASs with structure | CARO |
| Organism | organism mentions | taxonomy DBs |
| Organism subdivision | fiat parts of multicellular organism | CARO |
| Anatomical system | ASs of multiple organs | CARO |
| Organ | ASs of multiple multi-tissue structs. | CARO |
| Multi-tissue structure | AS of multiple tissues | CARO |
| Tissue | ASs of similar cells and ECM | CARO |
| Developing anatomical structure | ASs varying in granularity due to development | CARO |
| Cell | ASs of cell compartment, surrounded by PM | CL |
| Cellular component | ASs that are parts of cells | GO-CC |
| Organism substance | gaseous, liquid or semisolid material ASs | CARO |
| Immaterial anatomical entity | anatomical entities without mass | CARO |
| Molecular entity | | |
| Gene or gene product | genes, RNA and proteins | gene/protein DBs |
| Simple chemical | simple, non-repetitive chemical entities | ChEBI |
| Protein domain or region | parts of protein molecules | |
| Amino acid | amino acid residues | ChEBI |
| DNA domain or region | short, specifically identified spans of DNA | |
| Pathological formation | pathological material organism parts | |
| Cancer | cancerous pathological formations | |

The above *simple* events in Table 2.1 are selected keeping in mind to test our proposed system on a diverse hierarchy of entities, and also across single, binart and multiple valued arguments.

## 2.4 Summary

Here we laid out an in-depth analysis of the Cancer Genetics task setting, and the purview of the task undertaken by us, and the reasons behind the same. We have dealt with the definition of the task components, replete with a healthy dosage of instances at appropriate place. We have also described the nature of the task in terms of techniques commonly used in NLP, which will be subsequently elaborated in later chapters.

# Chapter 3

# Previous Work

## 3.1 Participants

According to the CG Task website, 6 teams took part in the task and produced commendable systems offering upto 48% recall and 64% precision, using various methodologies to detect and classify event and its argument. The top 3 teams are listed in the table 3.1,

| Team Name | Institute | Recall | Precision |
|---|---|---|---|
| TEES-2.1 | University of Turku, Finland | 48.76 | 64.17 |
| NaCtTeM | University of Manchester, UK | 48.89 | 55.82 |
| NCBI | NCBI, US | 38.28 | 58.84 |

Table 3.1: Result of Top 3 Institutes

## 3.2 Methodologies Used

## 3.3 Discussion

### 3.3.1 Introduction

The methodology varied from team to team. While some used Rule-based information extraction approach, some degenerated the problem into subgraph matching, and then there were others who used SVM, Parsers and CRF based named NER tools for the purpose. We discuss two of the previous works by University of Turku, based on Support Vector Machine and a Rule-based approach, as implemented by RelAgent Private Ltd.

| **Team** | **Lexical** | **Syntactic** | **Trigger** | **Arg** | **Group** |
|---|---|---|---|---|---|
| TEES-2.1 | Porter | McCCJ<br>SD | SVM | SVM | SVM |
| NaCtTeM | Snowball | Enju<br>GDep | SVM | SVM | SVM |
| NCBI | MedPost<br>BLemm | McCCJ<br>SD | Joint Subgraph Matching | | |

Table 3.2: System Stack of Top 3 Institutes

### 3.3.2 TEES2.1 - University of Turku, Finland

**TEES** [Björne and Salakoski, 2013] is a support vector machine (SVM) based system that automatically extracted events and arguments from domain corpus . The system has been used successfully in previous event extraction tasks from BioNLP Community, and certain modifications were added, which derived task-specific event rules and constraints from the training data, to automatically adapt to a new corpora.

#### 3.3.2.1 Methodology

Turku Event Extraction System (TEES) - 2.1, uses machine learning to extract text-bound graphs between trigger words and entities, where they are the nodes of the graph and the edges between them define the relation and the type of arguments.
A SVM with a linear kernel has been used for performing multi-class classification, which sub-divides the complex task of generating graph into smaller constituents.

#### 3.3.2.2 Preprocessing

TEES converts the shared task format of *text, a1, a2* corpora into an internal XML format, where entities are nodes and both event arguments and binary relations are stored as edge elements. Event consists of a trigger node whose type is equal to the type of the event.
Tokenization of the corpus is done via a standard tokenizer, and McCCJ parser is used to generate parses, which are then converted into a collapsed Stanford dependency scheme.

#### 3.3.2.3 Extraction and Classification

TEES follows three primary steps to detect presence of trigger words.

- *Firstly*, trigger words are detected by classifying each non-named entity token (un-annotated words) into one of the trigger classes or as negative. Only the valid edges in the graphs are stored based on provided information about entity types.

- *Second*, for each trigger word node and for each of its outgoing edges signifying a relation with an argument, an *unmerging* example is generated, and classified as a true event or not, which helps separate the overlapping events into valid single entity, for generating in the format as desired of BioNLP ST.

- *Lastly*, the multi edge path to arguments of arguments are collapsed in case of *site* type arguments, and are collapsed, via unified site-argument representation thus linking the argument of *site* type argument, directly to the event itself.

For events that can have modifier, another step is added to detect the same.

#### 3.3.2.4 Pitfalls

As mentioned in their paper, the current implementation of the automated annotation scheme learning system results in invalid event structures being produced for that have multiple argument types of binary value, it fails in recognizing when atleast one of the argument type must be present, and would require additional learning rule for arguments which are mandatory. An ad-hoc heuristics was implemented to overcome the error temporarily.

## 3.4 Summary

Although an in-depth discussions of all the techniques observed in this task setting is well out of the scope of this discourse, due to the sheer paucity of space. It is indeed interesting to observe that all the system that has been developed for this task, entails a lot of post-processing in order to ensure a high precision. As will be seen later, we will also indulge in post processing, but on a much lower scale and more intrinsic to the definition of the task, to achieve a same improvement.

# Chapter 4

# Current work

## 4.1 Task Undertaken

As mentioned previously, the scope of our task is limited to

- **Identifying Trigger Words for Events**
- **Identifying and Classifying Arguments**

## 4.2 Methodology

The methodology for building an automated system for information extraction is described, while the implementation of the said methodology with parameter selection are defined in a later section.

### 4.2.1 Introduction

The provided documents are first preprocessed. This is done so as to arrive at a better *word embedding in vector space* using an implementation of word2vec [Mikolov et al., 2013a] by Radim Rehurek [Rehurek et al., 2011], which will be dealt with subsequently. Following the word embedding, we use Recurrent Neural Network (RNN) and Long Short Term Memory Network (LSTMN) implementation [Schaul et al., 2010], for sequence labelling on the entire corpora to detect the trigger word. Then again the corpus is preprocessed, and a second word embedding is obtained, however this time with a more specific set of boundaries, capturing the limit upto which the arguments of an event may be present. Then we use both RNN and LSTMN, to obtain the the arguments of a specific type for a specific event in the corpus.

Here we digress a bit to discuss about the external concepts and their implementation we have incorporated in our methodology along with the reasons for incorporating them,

### 4.2.2 Word Embeddings via Distributed Representation

This follows from Neural Network Language Modelling. Obtaining the meaning and relations between the words in a corpus like a human does, remains the *holy grail* of the Natural Language Processing domain. Although achieving a system that does that have thus far eluded the community, there have been developments that have instrumental in mining the semantic relations between word based upon their context.
Neural Network based Word Embedding Language Model (also called Probabilistic Neural Language Model) approaches have been proposed over a decade back [Bengio et al., 2003], where the words in a corpus are represented as feature vector of dimension $D$ of real valued numbers. The *feature vector* associated with a word, is a point in the $D$-dimensional space, sharing a relation such that the vectors of words frequently co-occurring in the same context are similar, and offers a host of other such operations.For example,

$$vec_{king} \text{ - } vec_{man} + vec_{woman} \rightarrow vec_{queen}$$

Suppose the phrases *brave as a tiger* and *brave as lion* appears in a corpus for a couple of times. On subsequent training, vectors will be yielded where,

$$vec_{lion} \textit{ is very close to } vec_{tiger}$$

The advantage of these architectures is that they learn an embedding for words in a continuous vector space, with aforesaid set of features to name a few, that helps to provide good generalization even when the number of training examples is insufficient [Morin and Bengio, 2005]. Given the context of a word, the neural network architecture tries to compute the probability of the candidate word, and adjust its feature vector accordingly.

*Word2Vec*[Mikolov et al., 2013b], published by Google in 2013, is a (shallow) recurrent neural network (RNN) implementation that learns *distributed representations* for words. The advantage of Word2Vec over previous models, based on the same premise, is in terms of the time taken for learning the word embeddings,. which is considerably less and comparable to models based on *n-grams*.

#### 4.2.2.1 Methodology of Word2Vec

They proposed a *skip-gram with negative sampling* (SGNS) model, by trying to maximize the dot-product of the $D$-dimensions feature vector of frequently occurring word and its context which is also represented as $D$-dimensional vector, limited by the window size, and minimize the dot-product for the pairs outside of the context-window. For example,

....the National Animal of India is Tiger....

Let us suppose the candidate word $W$ is *Tiger* and lets say we are considering a right-sided window of size 6, hence its context $C$ will be *the National Animal of India is.* The SGNS method tries to maximize the dot-product of the $D$-dimensional feature vectors of $W$ and $C$, while trying to minimize for $W$ and other context sampled randomly out of the remaining context (hence negative sampling). It has been shown that SGNS achieves its objective, when

$$\text{vec}_W \cdot \text{vec}_C = PMI(W,C) \text{ - } \log k$$

where $k$ is the number of negative samples, and $PMI$ is the point-wise mutual information between candidate word $W$ and context $C$, and hence the system *degenerates to factorizing a shifted PMI Matrix*, where the shift value is log $k$, therefore learning the values of the real-valued feature vectors of the words in the corpus. The objective is trained in an online fashion using stochastic gradient descent and back-propagation updates over the observed set of $W$ and $C$ pairs in the corpora, in such a way, if two words $W_1$ and $W_2$, that happens to appear in similar context $C$,not necessarily together, will have similar or near-similar word embeddings [Levy and Goldberg, 2014].

#### 4.2.2.2 Application Scope in our Task

The idea from this discourse that would be later utilized for the aim of our task is that,

- **Identifying Trigger Words** : The Context surrounding a would-be trigger word for a particular event, in training data and test data will be similar, and hence they trigger word for a particular event in both the corpus will have a near equivalent feature vectors.
- **Identifying Arguments** : Similar arguments can be extended to the arguments as well, where the surrounding context containing the description of the candidate event will be instrumental, in determining the actual arguments from scores of argument candidates.

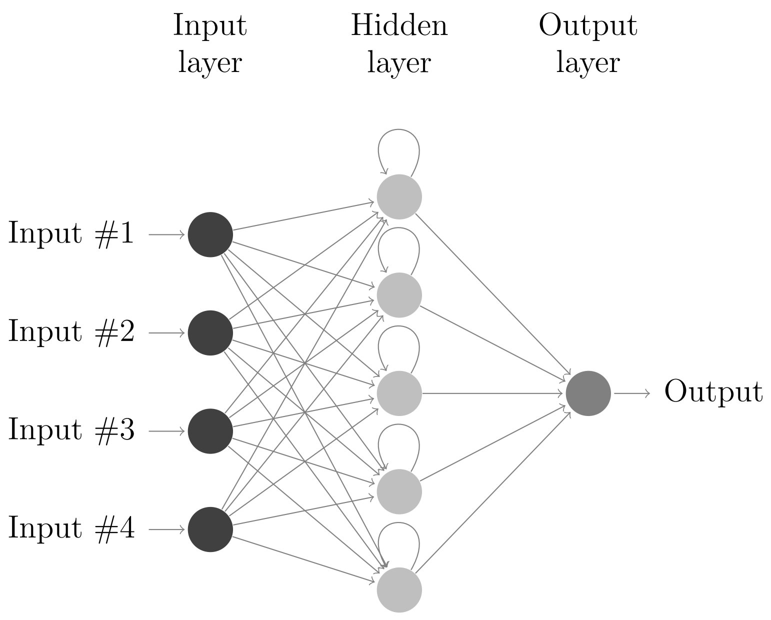
### 4.2.3 Recurrent Neural Network assisted Information Extraction

Artificial Recurrent Neural Networks (*aRNN*) is just like an ordinary Artificial Neural Network (or Multi-Layer Perceptron), composed of neurons, which are the computational unit. However besides having connection from lower layers to upper layers, it may have connection from one layer to itself (called *self loop*) and from upper layers to lower layers (called *back loop*), commonly known as *feedback loop*.

#### 4.2.3.1 Limitations of Conventional Neural Network

One of the glaring limitations of the normal neural network is that the mapping of input vectors $v_I$, the feature vectors, to output vectors $v_O$, which might represent probabilities of different class, uses only a fixed number of computation steps, which is proportional to the depth of the network. When presented with one $v_I$, it produces $v_O$ with the help of its weight, and then proceeds to do the same for the next $v_I$, in a similar fashion. These network does not store any *state*, in the sense that the observed $v_I$ can in no way have effect on the output of succeeding $v_I$. Hence they are unable to perform sequence classification in general. Although they are good at classification, this shortcoming of theirs leave them quite handicapped in performing various NLP tasks such as POS-Tagging, Named Entity Recognition and so on, which requires the system to possess a state that is modified in response to the current $v_I$.

Figure 4.1: Example of a RNN with self feedback loops in the hidden layer.



#### 4.2.3.2 Dynamical Nature of RNN

The requirement of an aRNN is that it must have atleast one type of feedback loop, so that the activation can flow round in a loop from previous time step. This enables the neuron to maintain a *state* of some sort, which is dependent on the cumulation of the activation of the prior inputs, for that particular neuron in the network. This values of this activation feedback loop changes at each time step based on the updation rule of the learning of the network, and is called the *Short Term Memory.* All this *state* for individual neuron having a feedback loop, culminates in the *state* of the network. This enables the network to do *temporal processing* and *learn sequences*, like performing sequence recognition/reproduction.

A single layer *Recurrent Neural Network* , as depicted in $Fig.$4.1, is defined as a single layer MLP, with the previous set of hidden unit activations feeding back into the network along with the input layer activation. The self feedback loop provides a delay of one time unit. Lets say for such a network, $x(t)$ define the input layer activation at a time $t$, similarly $h(t)$ define the hidden layer activation at a time $t$ and the output layer by $y(t)$. Let us define the input layer activation function, hidden layer activation function and output layer activation function as $f_I()$, $f_H()$ and $f_Y()$, respectively. Let us also define the input layer to hidden layer weight matrix , hidden layer to hidden layer weight matrix and hidden layer to output layer weight matrix as $W_{IH}$, $W_{HH}$ and $W_{HO}$, respectively.

$$\begin{aligned} h(t) &= f_H(W_{IH}x(t) + W_{HH}h(t-1)) \\ &= f_H(W_{IH}x(t) + W_{HH}f_H(W_{IH}x(t-1) + W_{HH}h(t-2))) \\ &. \\ &. \end{aligned} \tag{4.1}$$

Using the above expansion, we have

$$\begin{aligned} y(t) &= f_O(W_{HO}h(t)) \\ &= f_O(W_{HO}f_H(W_{IH}x(t) + W_{HH}h(t-1))) \\ &= f_O(W_{HO}f_H(W_{IH}x(t) + W_{HH}f_H(W_{IH}x(t-1) + W_{HH}h(t-2)))) \\ &. \\ &. \end{aligned} \tag{4.2}$$

Thus we see at time $t$, the output of the network is dependent upon the the *short term memrory state* of the system $h(t-1)$, which in turn is influenced by $x(t-1)$ to $x(1)$ as can be seen from the recurrence relation. Hence we can say the output at $t$ is influenced by $x(t)$ to $x(1)$, in other words all the previous inputs to a varying degree, with generally a diminishing effect as one trace backs to the first input observed at the network.

The above two equations, help express the recurrent neural network as a *dynamical system*. In general, the state of a dynamical system is a set of values that summarizes all the information about the past behaviour of a system, which is instrumental in producing its behaviour in the futures, apart from the effect of an input. Thus, in this case the hidden unit activations from the previous time step defines *state*.

Besides the short term memory state (so called because it updates at each time step), we also have a *Long Term Memory State* encoded by the networks in the form of the weights of the network, especially $W_{HH}$, which changes but on a slower timescale than the activations.

The Universal Approximation Theory [Csáji, 2001] states that,

*Any non-linear dynamical system can be approximated to any accuracy by a recurrent neural network, with no restrictions on the compactness of the state space, provided that the network has enough sigmoidal hidden units.*

#### 4.2.3.3 Limitations of RNN

While all seems fine with the Recurrent Neural Network model, it suffers from a drawback because of the diminishing effect of the inputs, which are farther into the past. This is known as Short Term Memory Effect. That is to say, an input from the observed input sequence which ought to have played a vital role for classifying the present input, will have a much diminished effect owing to the repeated multiplication of $W_{HH}$ with its vector. Weights less than 1.0 will exponentially reduce the activation, weights larger than 1.0 will cause it to increase. The non-linear activation functions of the hidden units will hopefully prevent it from growing without bound.
Thus for a task with *long term dependencies*, where the the desired output at certain time depends on the input at time much earlier on the time scale, conventional aRNN falters. Experiments [Bengio et al., 1992] have show that parameters hovers in a suboptimal solution space, because of taking only short term dependencies into account, and not the long term ones.

### 4.2.3.4 Long Short Term Memory Networks (LSTMN)

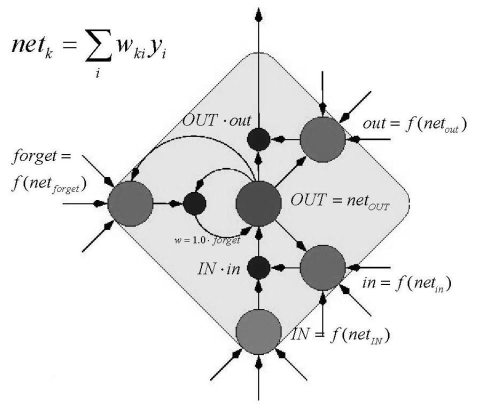


Figure 4.2: A Memory Cell of LSTM Network.

Typically, gradient based RNNs cannot reliably use information which lies more than about 10 time steps in the past [Hochreiter and Schmidhuber, 1997]. To address this problem, we will be additionally looking into another model known as Long Short Term Memory [Hochreiter and Schmidhuber, 1997]. In these networks in place of usual neurons in the layers, we have *memory cell*, as show in Fig. 4.2. At the core is a linear summing unit. At any given time step, it aggregates the combined input it receives. Its self recurrent connection has a fixed weight of 1.0 (which prevents exponential decay of activations from way back in the past, thereby solving the drawback of conventional RNNs). The *input gate IN* modulates whether the input from other cells or layer of cells are to be fed to the linear unit at the core , thereby modulating the effect of particular input vectors on the the *state* of the cell, and similarly there is an *output gate OUT* that modulates whether the output of the liner unit is to be broadcasted to the network, thereby modulating the effect this cell has on the output of the network for particular input vectors. The fixed weight of self recurrent connection can be set to 0 via the gate on the left hand side , known as the *forget gate* [Gers et al., 2000], thereby resetting the *state* of the cell at appropriate times, which helps the cell behave like a *counter* for certain application and also helps it from being influenced by inputs very long back in the past that does not have bearings on the present input.

LSTM Networks has been used in past successfully for sequence learning and classification problem with great success, such as Handwriting Recognition [Graves et al., 2008], Speech Recognition [Graves et al., 2013], Rhythm Learning [Gers et al., 2003], etc.

### 4.2.3.5 Application Scope in our Task

The capability a RNN/LSTM network to perform sequence classification, is exactly what will be required to extract trigger words as well as arguments of a given type for an event.

Usage of Recurrent Neural Networks for performing sequence classification tasks specific to Natural Language Processing Domain has been limited to application based on feature vectors derived of *syntactic* origin, rather than *semantic* origin, such as parsing [Chen and Manning, 2014], word sense disambiguation [Veronis and Ide, 1990],etc. It is only of late that people are using semantic embeddings of word, derived from distributed representations implementations, like word2vec,GloVe, etc for these purposes.

Moving forward, we describe in details the steps for arriving at the word embedding of the given corpora, and the parameters for that. Then we discuss the topology of the RNN/LSTM architectures we use for the sequence classification job, and the parameters associated with them as well as the training process used.

### 4.2.4 Implementation for Trigger Word Extraction

The documents provided by the Cancer Genetics Task, for training and development purposes, can be broadly classified into 3 types,

- *.txt* files containing abstracts of scholarly articles related to cancer genetics.
- *.a1* files containing the annotation of entities present in the corresponding text files.
- *.a2* files containing the annotation of events along with their arguments, equivalences and modification, that needs to be extracted.

#### 4.2.4.1 Processing of Stop-Words and Punctuations

Stop-Words are **not**removed in order to facilitate better semantic representation through word embedding, as the algorithm we will be using later on for word embedding, relies on the broader context of the sentence in order to produce high-quality word vectors, as will be observed later. Further more, all the punctuation symbols are removed, as they in any way do not influence the word embedding algorithm, and also it will facilitate us later to effectively tokenize for word substitution.

#### 4.2.4.2 Substitution of Entity Names

**Hypothesis 1 (Trigger Word & Event Specific Argument Collocation)** *The trigger words representing a certain event will have a high collocation with those types of arguments, as are available for the concerned event, in both the training and the test corpus, and hence these trigger words in the training and the test corpus will have a high* similarity*.*

Given the entity annotations in the .a1 files, we substitute the actual word representing a said entity with its entity class as our first approach. For example,

....presence of **aflatoxin**$_{SimpleChemical}$ *causes* the *growth* of **tumors**$_{Cancer}$ in **Liver**$_{Organ}$....

becomes

....presence of **SimpleChemical** *causes* the *growth* of **Cancer** in **Organ**....

This is done in order to facilitate word embedding algorithm to provide semantically meaningful embeddings for the would-be trigger words, based on their collocation with the word from their available argument type. This is done because the *actual* name of the arguments are few and rare, and in most of the case does not occur more than once, therefore substituting the entities with their type names provide *homogeneous* context, for the word2vec.

#### 4.2.4.3 Generation of Word Embeddings via Word2Vec

The substituted corpus is then used for generating word embeddings, via word2vec. Window sizes used were in {8,12,16,20,24}. For training method we use *skip-gram*, coupled with *hierarchical softmax*, instead of *negative sampling* because of the more emphasis needed to be given on the infrequent words, viz trigger words. Learning rates were kept at default. The no. of iterations were varied from {15,25,50,100}

#### 4.2.4.4 Training with RNN/LSTM Network

Architectures having partially recurrent connection, that is having full recurrent connection only in the hidden layers, were used. The networks were made upto 2 layers deep. The input layers were *Linear* and the output layers were *Softmax*. For simple RNN, the hidden layers were composed of {150,200,300} *Tanh* nodes, while for LSTM network they were composed of {150,300} *LSTM memory cell*. PyBrain [Schaul et al., 2010] has been used for implementing the network.

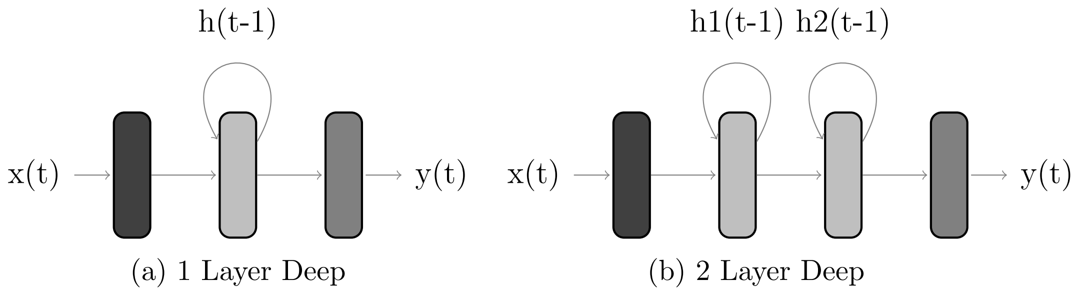


Figure 4.3: Topology of the networks used for trigger extraction

For training the networks, we use a variation of *Back Propagation through Time*, called *Resilient Back Propagation* (RProp). Advantage is in faster training per epoch, and parameters such as momentum or learning rate need not be mentioned. We do a one-vs-rest sequence classification.

It must be noted that a separate setup is also implemented for extracting trigger words, based on a conditional random field (CRF)-based named entity recognition tool, for comparison.

#### 4.2.4.5 Minimization of False Positives

It was seen from the training Corpus, that trigger words are composed of only two types of POS, noun and verb. For example,

...the$_{art}$ *expression*$_{noun}$ of$_{prep}$ gene$_{noun}$...
or
... aflatoxin$_{art}$ *inhibits*$_{verb}$ repair$_{noun}$ enzymes$_{noun}$...

Thus, the tokenized corpus is fed to Genia tagger, for POS tagging and the output for the test corpus in post-processed accordingly. This helps in increasing the Precision of the task.

### 4.2.5 Implementation for Argument Extraction

Following identification of trigger words, we preprocess the text further for argument extraction and then subsequent word embedding and training.

#### 4.2.5.1 Substitution of Entity and Event Names, and preservation of Prepositions

The hypotheses being put forward is similar to the one that has been put forward for trigger word extraction.

**Hypothesis 2 (Event Specific Argument & Trigger Word Collocation)** *The arguments of a particular type(s) for an event will have a high collocation with the trigger words for those event, in both the training and the test corpus, and hence these trigger words in the training and the test corpus will have a high* similarity*.*

Given the entity annotations in the .a1 files, we substitute the as before, while also switching out the trigger word for a particular event by its event name. For example,

....presence of $\textbf{aflatoxin}_{SimpleChemical}$ $causes^{Regulation}$ the $growth^{Growth}$ of $\textbf{tumors}_{Cancer}$ in $\textbf{Liver}_{Organ}$....

becomes

....presence of **SimpleChemical** *regulation* the *growth* of **Cancer** in **Organ**....

**Hypothesis 3 (Argument and Type specific Preposition collocation)** *The Argument and the preposition which defines its type, will have a **very** high collocation.*

In the above example, the prepositon *of* denotes the presence of **theme** type argument, whereas *in* denotes the presence of **atLoc** type argument.

The reason for substitution is same as before.

#### 4.2.5.2 Corpus Truncation

Suppose we want to figure out the arguments for an event $E$ that has arguments of type $T1$ and $T2$. Instead of using the whole substituted corpus as training, in order to provide the word2vec tool with more specificity, we use a truncated corpus as following:

- *Sentence Selection*: Following a quick processing of the training corpus, it is found that for all of the event types we have considered, *all the arguments of the event lies within the same sentence in 92% of the case*, with the percentage being higher for event having single argument and lower for events having multiple arguments, of same or different type.*The percentage goes upto 100%,if we include both the previous and the next sentence.* Therefore in order to capture the the arguments of a given sentence, it suffices to search only within these given bound, and truncate the corpus accordingly. In these case, we have limited ourselves to the sentence containing the event.

- *Division and Substitution*: For the event $E$, we divided the truncated into two part, one for extracting arguments belonging to type $T1$ and another for the type $T2$, i.e We divide the task based on the no. of types of arguments. Although we prepare the word embeddings together, training by neural network is done separately for arguments of different types.

#### 4.2.5.3 Generation of Word Embeddings via Word2Vec

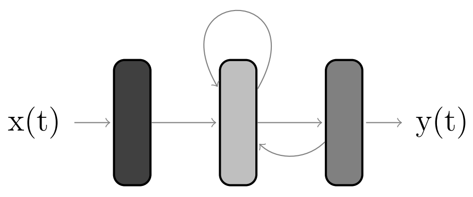
The truncated and substituted corpus is then used for generating word embeddings, via word2vec as before with skip-gram and hierarchical softmax.

#### 4.2.5.4 Training with RNN/LSTM Network

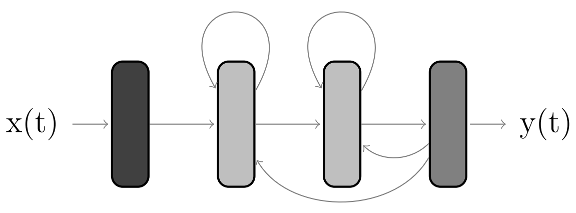
The constitution of the layers remain same as in the case for trigger word extraction. In addition to them, *Fully Output Recurrent* networks of 1 Layer and 2 Layer deeps were used for conventional RNNs, as show in Fig. 4.4.

#### 4.2.5.5 Minimization of False Positives

We use a post-processing heuristic to minimize the false positives, whereby we check only those possible argument candidates which belong to the valid hierarchy for the type of the argument of the current event being considered. For example, we only consider *entity* of *cell* hierarchy, while checking for *theme* type argument of *Cell Tranformation* event.



(a) 1 Layer Deep



(b) 2 Layer Deep

Figure 4.4: Topology of the extra networks used for argument extraction

## 4.3 Result

Here we discuss the result obtained for both parts of the task, using the aforesaid methodology, and compare it to the state of the art system as of now.

### 4.3.1 Benchmarking Metric

Although recall and precision @ K, are good parameters for judging the effectiveness of a system, we would be using the following metric:

- **Highest Recall** and the precision at highest recall.
- **Highest Precision @ 50% recall**, if reached
- **Highest Precision @ recall of state of the art**, if reached.

Further, we will be using graphical representation for *Cell Transformation*, which has both argument of simple type and complex type, for showing the performance of various topologies over iterations.

For the Argument Extraction part of the task, the precision of the system is not clearly defined by the Cancer Genetics Task, and hence we deal with only recall.

Further note that the results from Networks with 2 hidden layers, have not been published due to their performance not meeting the expectations, and have henceforth been ignored. The possible reason for such unexpected performance from these networks have been discussed in later sections.

### 4.3.2 Trigger Word Extraction

#### 4.3.2.1 Cell Transformation

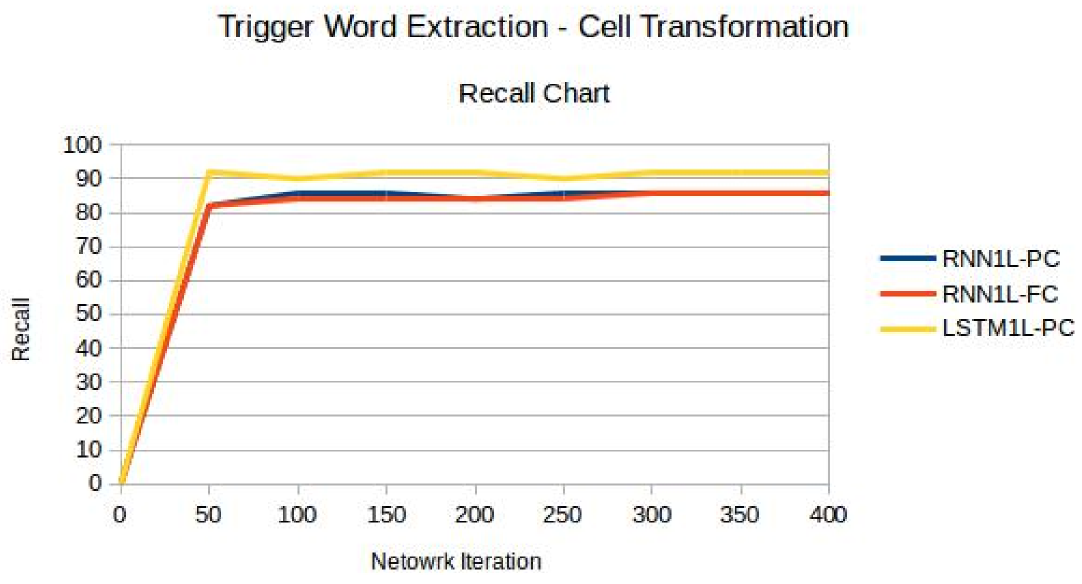


Figure 4.5: Gene Expression - Trigger Word Extraction Recall Graph

- **RNN1L-PC** 150 nodes in Hidden Layer. Word2vec model with window size 8, and 25 Iterations.
- **RNN1L-FC** 150 nodes in Hidden Layer. Word2vec model with window size 12, and 50 Iterations.
- **LSTM1L-PC** 150 nodes in Hidden Layer. Word2vec model with window size 16, and 50 Iterations.

| | **RNN1L-PC** | **RNN1L-FC** | **LSTM1L-PC** | **CRF based NER** |
|---|---|---|---|---|
| **Recall(R)** | 86 | 86 | 92 | 96 |
| **P @ R** | 57 | 61 | 66 | 93 |

Table 4.1: Highest Recall and Precision - Cell Transformation

| | RNN1L-PC | RNN1L-FC | LSTM1L-PC | CRF based NER |
|---|---|---|---|---|
| **P** | 74 | 80 | 82 | 98 |

Table 4.2: Precision @ 50% Recall - Cell Transformation

**Highest Precision @ Recall of State of the Art**: Not Applicable

#### 4.3.2.2 Gene Expression

| | RNN1L-PC | RNN1L-FC | LSTM1L-PC | CRF based NER |
|---|---|---|---|---|
| **Recall(R)** | 75 | 81 | 84 | 95 |
| **P @ R** | 52 | 57 | 64 | 96 |

Table 4.3: Highest Recall and Precision - Gene Expression

| | RNN1L-PC | RNN1L-FC | LSTM1L-PC | CRF based NER |
|---|---|---|---|---|
| **P** | 68 | 72 | 79 | 98 |

Table 4.4: Precision @ 50% Recall - Gene Expression

#### 4.3.2.3 Blood Vessel Development

| | RNN1L-PC | RNN1L-FC | LSTM1L-PC | CRF based NER |
|---|---|---|---|---|
| **Recall(R)** | 66 | 64 | 75 | 91 |
| **P @ R** | 37 | 41 | 49 | 92 |

Table 4.5: Highest Recall and Precision - Blood Vessel Development

| | RNN1L-PC | RNN1L-FC | LSTM1L-PC | CRF based NER |
|---|---|---|---|---|
| **P** | 56 | 59 | 70 | 96 |

Table 4.6: Precision @ 50% Recall - Blood Vessel Development

**Highest Precision @ Recall of State of the Art**: Not Applicable

#### 4.3.2.4 Cell Death

| | **RNN1L-PC** | **RNN1L-FC** | **LSTM1L-PC** | **CRF based NER** |
|---|---|---|---|---|
| **Recall(R)** | 82 | 77 | 86 | 93 |
| **P @ R** | 51 | 58 | 64 | 95 |

Table 4.7: Highest Recall and Precision - Cell Death

| | **RNN1L-PC** | **RNN1L-FC** | **LSTM1L-PC** | **CRF based NER** |
|---|---|---|---|---|
| **P** | 67 | 72 | 79 | 98 |

Table 4.8: Precision @ 50% Recall - Cell Death

**Highest Precision @ Recall of State of the Art**: Not Applicable

#### 4.3.2.5 Performance across Events

| | **Cell Transformation** | **Gene Expression** | **Blood Vessel Development** | **Cell Death** |
|---|---|---|---|---|
| **Maximum Recall(R)** | 92(66) | 84(64) | 75(49) | 86(64) |
| **CRF based NER Recall** | 96(93) | 95(96) | 91(92) | 93(95) |

Table 4.9: Trigger Word Extraction Performance Comparison. Precisions are given in brackets

### 4.3.3 Argument Extraction

#### 4.3.3.1 Cell Transformation

Since the NER based Trigger Word Extraction outperforms our RNN based system for the same tasl, we incorporate the NER into our system stack and perform Argument Extraction based on its result.

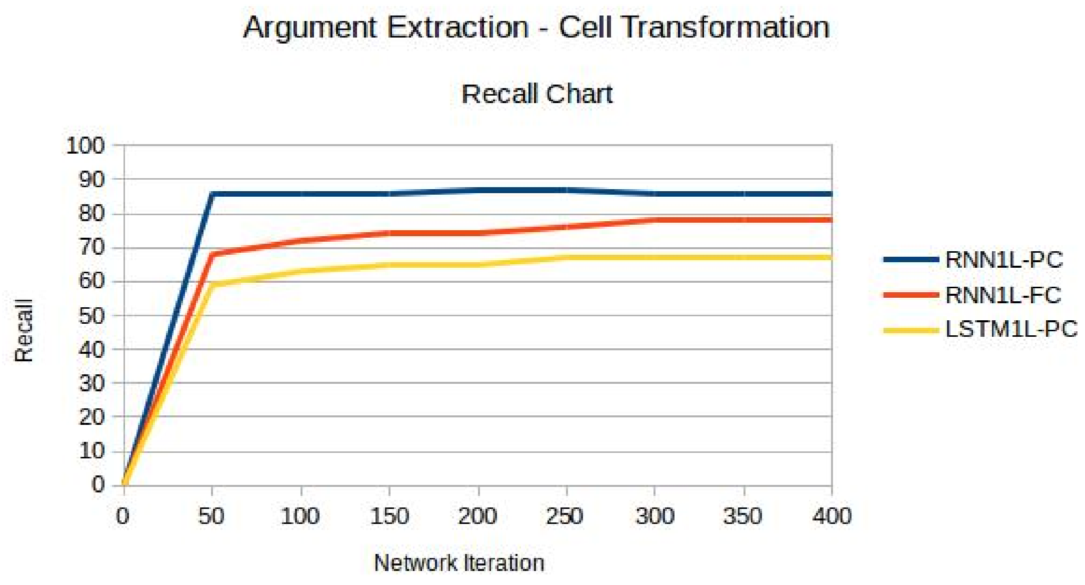


Figure 4.6: Gene Expression - Argument Extraction Recall Graph

- **RNN1L-PC** 150 nodes in Hidden Layer. Word2vec model with window size 8, and 50 Iterations.
- **RNN1L-FC** 150 nodes in Hidden Layer. Word2vec model with window size 8, and 50 Iterations.
- **LSTM1L-PC** 300 nodes in Hidden Layer. Word2vec model with window size 20, and 100 Iterations.

| | **RNN1L-PC** | **RNN1L-FC** | **LSTM1L-PC** | **TEES 2.1** |
|---|---|---|---|---|
| **Recall(R)** | 87 | 78 | 64 | 79 |

Table 4.10: Highest Recall - Argument Extraction - Cell Transformation

#### 4.3.3.2 Gene Expression

| | RNN1L-PC | RNN1L-FC | LSTM1L-PC | TEES 2.1 |
|---|---|---|---|---|
| **Recall(R)** | 58 | 60 | 65 | 76 |

Table 4.11: Highest Recall - Argument Extraction - Gene Expression

#### 4.3.3.3 Blood Vessel Development

| | RNN1L-PC | RNN1L-FC | LSTM1L-PC | TEES 2.1 |
|---|---|---|---|---|
| **Recall(R)** | 82 | 81 | 75 | 80 |

Table 4.12: Highest Recall - Argument Extraction - Blood Vessel Development

#### 4.3.3.4 Cell Death

| | RNN1L-PC | RNN1L-FC | LSTM1L-PC | TEES 2.1 |
|---|---|---|---|---|
| **Recall(R)** | 72 | 68 | 59 | 67 |

Table 4.13: Highest Recall - Argument Extraction - Cell Death

#### 4.3.3.5 Performance across Events

| | Cell Transformation | Gene Expression | Blood Vessel Development | Cell Death |
|---|---|---|---|---|
| **Our System** | 87 | 65 | 82 | 72 |
| **TEES 2.1** | 79 | 76 | 80 | 67 |
| **NacTeM** | 85 | 79 | 73 | 85 |
| **NCBI** | 75 | 72 | 79 | 80 |

Table 4.14: Argument Extraction Performance Comparison

### 4.3.4 Discussion

#### 4.3.4.1 Trigger Word Extraction

- It can be seen for the events *Cell Transformation*, *Gene Expression* and *Cell death*, the system shows appreciable recall when compared to the current state of the art. However that of Blood Vessel Development, languishes in terms of that parameter. Reasons for such observed behaviour can be reasoned as follows,
  The 3 better performing events have trigger words limited to certain inflectional forms of certain root words, however that is not the case for the Blood Vessel Development. For example,
  - For *Cell Transformation*, the trigger words are variations of one word such as *transformation*, *transform*, etc.
  - For *Gene Expression*, the trigger words are variations of the word *expression* .
  - For *Cell death*, the trigger words are variations of the word *death* or *die*.
  - However, for *Blood Vessel Development*, the trigger words include word across a wider spectrum. For eg, *development*, *formation*, *growth*, besides scientific terms like *angiogenesis* etc. which in addition to being diverse also collides with candidate trigger words for other events.
- The Precision across all the events are sub-par in comparison with the CRF-based NER methodology. Reason can range from the network being stuck in a local minima, to similarity between candidate trigger words across separate events, as preViously discussed in the case of *Blood Vessel Development*, since the very methodology of the current task depends on the *semantic* nature of words rather that *syntactic*.
- LSTM based network out performs both fully and partially recurrent RNN in the sequence classification job as expected, and that too with higher precision.

#### 4.3.4.2 Argument Extraction

- For events involving binary valued arguments of one or multiple types, such as *Cell Transformation, Blood Vessel Development and Cell Death*, our proposed system either surpasses or matches upto the current State of the Art techniques. The binary argument can be of any hierarchy, as observed. Hence it can be said that, the system generalizes well for binary valued arguments belonging to any hierarchical position.
- The system struggles to keep up with the event that takes in multiple valued argument, such as *Gene Expression.* Different architecture may be tried to resolve the issue, but the paucity of the data is a factor.
- LSTM based network falters for the Argument Extraction sub-task, which is not expected, except for the event *Gene Expression.* Reasons can vary from paucity of data due to *corpus truncation*, to getting stuck in local minima. *Gene Expression* being the third most cited event, retains large corpus even after truncation

#### 4.3.4.3 Poor Results of Deeper Networks

The unexpected result observed from deeper networks can be attributed to the paucity of data. Deeper Networks need more training data, to tune its weight parameters, which unfortunately for us is not the case, as we are limited only to 600 abstracts from scholarly articles. Addition of resources from other task could help.

## 4.4 Summary

Here in this chapter we presented the concepts behind our methodology, and implementation of the same. We discussed in details about the use of word embeddings via distributed representation, which made it possible for us to use RNN and LSTMN for training to perform the trigger word extraction and argument extraction from the CG corpus. We found that in the Trigger Word Extraction subtask, the system had a recall rate in vicinity of the state of the art. However the precision was below par, reason for such a result could depend on the *momentum* of the network. For the Argument Extraction subtask, the system had recall comparable and at times better than the state of the art, thereby hinting at further research or work being warranted for the usage of Neural Network in the Information Extraction domain.

# Chapter 5

# Conclusion and Future Work

In this report, we documented the result of using RNNs and LSTMNs to perform information extraction, based on word embeddings. We presented a detailed outline of the Cancer Genetics Task, its components, aims and why NLP technique is required for it. We then proceeded to discuss the methodology of the state of the art technique. Followed by which a description of our methodology, the implementation and concepts behind it. The results of our implementations were compared, and the observed behaviour was discussed. Our system fared well in some events, specially the one with binary-valued arguments, however did not fare upto the standard for multi-valued arguments. However not whole of the task could be evaluated. The portions of the task that was not considered are,

- *Complex Events* with free arguments, i.e events which can take any hierarchy of entities or even other events as its arguments, such as *Regulation, Planned Process*, etc. In order to extend our system for such events, we need to add another computation layer to our stack whereby after extracting arguments for all the *simple events*, we need to perform trigger word extraction for *complex events*, and after necessary preprocessing, perform argument extraction.

- Modification of events. In order to classify an event as a *speculation* or *negation* type, we need to add another computation layer to our system stack, whereby after extraction of all *simple* and *complex* events, we train RNNs and LSTMNs on the preprocessed corpus, to identify the modifications.

# Bibliography

[Bengio et al., 1992] Bengio, Y., De Mori, R., Flammia, G., and Kompe, R. (1992). Global optimization of a neural network-hidden markov model hybrid. *Neural Networks, IEEE Transactions on*, 3(2):252–259.

[Bengio et al., 2003] Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. (2003). A neural probabilistic language model. *The Journal of Machine Learning Research*, 3:1137–1155.

[Björne and Salakoski, 2013] Björne, J. and Salakoski, T. (2013). Tees 2.1: Automated annotation scheme learning in the bionlp 2013 shared task. In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pages 16–25.

[Burges, 1998] Burges, C. J. (1998). A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2):121–167.

[Chen and Manning, 2014] Chen, D. and Manning, C. D. (2014). A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, volume 1, pages 740–750.

[Csáji, 2001] Csáji, B. C. (2001). Approximation with artificial neural networks. *Faculty of Sciences, Etvs Lornd University, Hungary*, 24.

[Gers et al., 2000] Gers, F. A., Schmidhuber, J., and Cummins, F. (2000). Learning to forget: Continual prediction with lstm. *Neural computation*, 12(10):2451–2471.

[Gers et al., 2003] Gers, F. A., Schraudolph, N. N., and Schmidhuber, J. (2003). Learning precise timing with lstm recurrent networks. *The Journal of Machine Learning Research*, 3:115–143.

[Graves et al., 2008] Graves, A., Liwicki, M., Bunke, H., Schmidhuber, J., and Fernández, S. (2008). Unconstrained on-line handwriting recognition with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 577–584.

[Graves et al., 2013] Graves, A., Mohamed, A.-r., and Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 6645–6649. IEEE.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

[Levy and Goldberg, 2014] Levy, O. and Goldberg, Y. (2014). Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems*, pages 2177–2185.

[Mikolov et al., 2013a] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

[Mikolov et al., 2013b] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, pages 3111–3119.

[Morin and Bengio, 2005] Morin, F. and Bengio, Y. (2005). Hierarchical probabilistic neural network language model. In *Proceedings of the international workshop on artificial intelligence and statistics*, pages 246–252. Citeseer.

[Nédellec et al., 2013] Nédellec, C., Bossy, R., Kim, J.-D., Kim, J.-J., Ohta, T., Pyysalo, S., and Zweigenbaum, P. (2013). Overview of bionlp shared task 2013. In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pages 1–7.

[Pyysalo et al., 2013] Pyysalo, S., Ohta, T., and Ananiadou, S. (2013). Overview of the cancer genetics (cg) task of bionlp shared task 2013. In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pages 58–66. Citeseer.

[Rehurek et al., 2011] Rehurek, R., Sojka, P., et al. (2011). Gensimstatistical semantics in python. *NA*.

[Schaul et al., 2010] Schaul, T., Bayer, J., Wierstra, D., Sun, Y., Felder, M., Sehnke, F., Rückstieß, T., and Schmidhuber, J. (2010). Pybrain. *The Journal of Machine Learning Research*, 11:743–746.

[Veronis and Ide, 1990] Veronis, J. and Ide, N. M. (1990). Word sense disambiguation with very large neural networks extracted from machine readable dictionaries. In *Proceedings of the 13th conference on Computational* *linguistics-Volume 2*, pages 389–394. Association for Computational Linguistics.