Automatic Classification of Citation Function Using Deep Neural Networks

A thesis submitted for
B. Tech. Project
in
Computer Science and Engineering

by
Varun Rawal (13CS10059)

advised by
Prof. Pabitra Mitra

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

May 2017

Certificate

This is to certify that the thesis titled **Automatic Classification of Citation Function Using Deep Neural Networks** submitted by **Varun Rawal (13CS10059)** to the Department of Computer Science and Engineering is a bonafide record of work carried out by him under my supervision and guidance. The thesis has fulfilled all the requirements as per the regulations of the Institute and, in my opinion, has reached the standard needed for submission.

Prof. Pabitra Mitra
Professor
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur
May 2017

Declaration

I, **Varun Rawal** hereby certify that this report titled **Automatic Classification of Citation Function Using Deep Neural Networks**

- Is an original work and has been done by me under the guidance of my supervisor;
- The work has not been submitted to any other Institute for any degree or diploma;
- While writing the report I have conformed to norms and guidelines given in the Ethical Code of Conduct of the Institute;
- Whenever I have used materials (data, model, figures and text) from other sources, I have given due credit to them by citing them in the text of the report, giving their details in the references, and following fair use doctrine policies of copy righted materials if any used in this thesis.

Varun Rawal (13CS10059)

$2^{nd} May$ 2017

Acknowledgement

I sincerely thank **Prof. Pabitra Mitra**, Department of Computer Science And Engineering, IIT Kharagpur for continuously supporting me throughout the duration of the project and providing me with valuable inputs. The report has been made under his guidance and support. His critical inputs helped me immensely in this project. I would like to extend my most sincere thanks to Prof. Pabitra Mitra.

Varun Rawal (13CS10059)

$2^{nd} May$ 2017

Abstract

The main aim of the project is to develop a Machine Learning framework based on Deep Neural Network learning architecture which is capable of automatic classification of the Citation Function of the given citation, i.e. given the citing paper and the specific paper cited by the former. The prediction of the Citation Function entails two major facets: to label the Citation Sentiment polarity as well as to anticipate the type of the Intent of the author to cite the given manuscript. Since any Deep Learning-based framework cannot be trained on a small-sized data corpus, it was eventually decided to choose the self-generated corpus (using the annotator-interface) for testing phase whilst we trained the model with the publicly available corpus faceted with citation sentiment only. In order to train the machine learning model, huge amounts of training datasets were needed, so the preliminary phase of the project focused on collection of adequate data corpus via development of a Crawler and a JAVA-based manual Annotator Utility Tool accompanied with a GUI Interface to enable speedy annotation of the citations. In this second phase of the project tenure, we deployed two well-known Deep Learning Architectures to classify the citation sentiments namely Recurrent Neural Networks (RNNs) based Long Short-Term Memory Networks (LSTMs) and Text Convolutional Neural Networks (Text - CNNs). The statistical visualizations on various learning metrics and backpropagated gradients of weights and biases at each layer were captured for both the neural learning architectures (LSTMs and Text CNNs). These visualizations were obtained via the help of Google-facilitated TensorFlow library's built-in utility tool, Tensorboard. We look forward to solving Automatic Citation Analysis (ACA) task in the

directions of using different corpora, different annotation schemes, different feature sets and different classifiers. The mentioned work is intended to be extended by attempting the deployment of Neural based Attention Learning Model Architectures as well as more adaptive Recursive Neural Networks taking up a universally acceptable annotation corpus at hand as the benchmark. The similar experiments can be performed for lots of feature vector variants like multi-word (n-gram) lexical features, linguistic features, structure features, location features, frequency features, sentiment features and many more.

TABLE OF CONTENTS

Chapter 1

Introduction

Publicizing citations are nice way to recognize the past related work in the field of domain the manuscript is being drafted in. The authenticity and justification of citation text has always been of keen intertest among researchers for discourse analysis, information science and library sciences for decades [7]. Part of this sustained interest in citations can be explained by the fact that bibliometric measures are commonly used to measure the impact of a researcher's work by how often they are cited.

Citations are relations between the cited and the citing articles and constitute a crucial content in publicizing literature. The automatic recognition of the rhetorical function of citations in scientific text has many applications, spanning from improvement of impact factor calculations to text summarisation and more informative citation indexing tasks. Citation function is defined as the author's reason for citing a given paper (e.g. acknowledgement of the use of the cited method).

1.1 Problem Objective

Many annotation schemes for encouraging citation works have been created over the years, and the question has been studied in detail, even to the level of in-depth interviews with writers about each individual citation. Following one of the reliable annotation scheme for citation function, this project aims to present a supervised machine learning framework to automatically classify citation function, which uses several shallow and linguistically-inspired features. We also extracted from the citation texts. We also observe a strong relationship between the two facets of describing citation function i.e. type of Citation Intention / Reason for Citation and the Citation Sentiment involved and that the classification of one would definitely help us in that of the other.

1.2 Citation Function – A Formal Definition

One may choose to draw a simple analogy from a basic mathematical function to the ***"Citation Function"***. Consider, for example, a simple mathematical function, say

$$f(x)\colon \mathbb{R} \to \mathbb{Z}$$

which simply maps all real numbers to integers. Here, $\mathbb{R}$ is the domain of the function and $\mathbb{Z}$ is the co-domain.
Similarly, a Citation Function Ω can be defined as a function which has as its domain, the set of Citing Papers and the set of Cited Papers as its Co-domain.

Thus, the Citation Function Ω maps a Citing Paper to all of its Citations (the papers it "Cites" / "Refers to".

Diagrammatically,

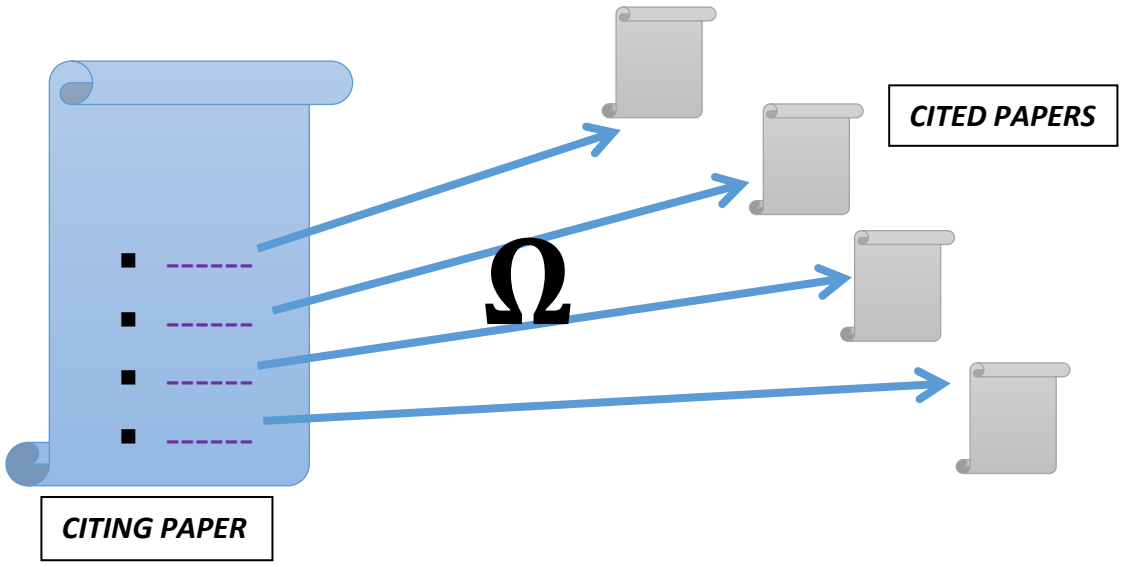


Fig 1.1 Representation of *Citation Function*

1.2.1 Citation Function: Two Primary Facets

According to the definition we've adopted, it encompasses the following two aspects of a Citation Link:

1. ***Citation Sentiment:***
    a. Positive
    b. Neutral
    c. Negative

    Classification of Citation Sentiment entails identification of the polarity of opinions, sentiments, emotions and attitudes expressed in text.

2. ***Citation Intent / Citation Type:***

    Can be defined in several different kinds of ways, according to the demand of the scenario, but in each case, one finds a **strong relationship** with the *Sentiment polarity involved*.

1.2.2 Categorization of Citation Function

Now that we have formally defined the citation function, it is required to provide a notion or a basis for characterization of the different types of Citation function that can possible exist with a given Citation Link.

Referring to the prior work, one can find several different varying bases of classification which characterize Citation Function as different combinations of Sentiment along with the Type of Author's Intention to cite.

Here, we summarize three such different possible classifications of Citation Function as Tables:

In [15], the scheme (Fig. 1.2) has been adapted from Spiegel-Rusing's (1977) after an analysis of a corpus of scientific articles in computational linguistics.

Sentiment Category	Description of Intention / Reason of Citation
Neutral	Contrast/Comparison in Goals or Methods
Neutral	Contrast/Comparison in Results
Neutral	Neutral description of cited work, Not enough textual evidence for any of the categories, or unlisted citation function
Negative	Author's work is stated to be superior to the cited work, Highlighting weaknesses of the Cited Work
Positive	This citation is positive about approach used or problem addressed (used to motivate work in current paper)
Positive	Author uses cited work as basis or starting point
Positive	Author borrows, adopts or modifies tools/algorithms/data/definitions
Positive	Author's work and cited work are similar / compatible/ provide support for each other

Fig 1.2 Annotation Scheme – I [15]

In [3], the scheme (Fig 1.3) narrowed down the categories to avoid overlapping nature of any two categories.

Sr No.	Description of Intention / Reason of Citation	Sentiment Involved
1	Continuing a Research from point where Cited paper finished.	Positive
2	Citing paper to use its ideas, definitions, terms in the Research work.	Positive
3	Citing a paper to refer to data it used and to draw similarities from the Data used.	Positive
4	Citing Paper to adopt part/full methodology the cited one adopted for a certain task.	Positive
5	Citing paper verified/proved a statement or enlightens with its details.	Positive
6	Citing Paper evaluated positively.	Positive
7	Ongoing Research giving proof of statement in Cited Paper.	Positive
8	Cited Paper provides historical facts regarding undergoing Research Problem.	Neutral
9	Citing a paper to refer to data also used in Current Research.	Neutral
10	Citing Paper contains Data and Material used throughout different phases.	Neutral
11	Citing Paper evaluated negatively.	Negative
12	Ongoing Research giving rebuttal of statement in Cited Paper.	Negative
13	Giving a new interpretation to the findings/statements in Cited Paper.	Negative

Fig 1.3 Annotation Scheme – II [3]

In [12] (Fig 1.4), the categories do not account for and are not so strongly correlated with the citation sentiment polarity. Instead, they just label the different categories which basically distinguish the citation types based on whether the citation shows other researchers' theories or methods for theoretical basis, or if it points out to the problems or gaps in related works.

Sr. No.	Intent of Citation	Type
1	Paying Homage to Pioneers	3
2	Giving Credit for Related Work	3
3	Identifying methodology, equipment, and so on…	1
4	Providing background reading	1
5	Correcting one's own work	3
6	Correcting the work of others	3
7	Criticizing Previous Work	3
8	Substantiating Claims	1
9	Alerting to forthcoming work	3
10	Providing leads to poorly disseminated, poorly indexed, or uncited work	3
11	Authenticating data and classes of fact – physical constants, and so on..	3
12	Identifying original publications in which an idea or concept was discussed	3
13	Identifying original publications or other work describing an eponymous concept or term	3
14	Disclaiming work or ideas of others (negative claims)	2
15	Disputing priority claims of others (negative homage)	2

Fig 1.4 Annotation Scheme – III [12]

1.3 Deploying Neural Network Architectures

In this second phase of the project tenure, we deployed well-known Deep Learning Architectures to classify the citation sentiments. The motivation behind deploying two primary architectures for the classification of the Citation function were:

1.3.1 RNN – based LSTMs

Recurrent Neural Networks (RNNs) are popular models that have shown great promise in many NLP-related tasks. A variant of RNN-based architectures designed especially for long-term-dependency-critical tasks like sentiment classification, called the ***Long Short-Term Memory Networks (LSTMs)*** has been shown to outperform almost every other RNN-based model [6].

Once the **Sentiment** is classified with high confidence, the task of classifying the **Type of Citation Intent** is expected to require relatively less effort as it bears a Strong Relationship with the Sentiment involved.

1.3.2 Text CNNs

Convolutional Neural Networks (CNNs) have gained immense popularity in achieving state-of-the-art classification accuracies engaged in machine learning tasks involving images. However, Yoon Kim first published a work demonstrating the way the ability of CNNs can be further utilized to extend their classification power to tasks involving text datatypes too [8]. The architecture popularly known as Text CNNs have been implemented by several researchers since then. They have been deployed in wide applications for machine learning tasks of image and video recognition, recommender systems as well as natural language processing.

Chapter 2

Datasets

Due to the lack of any standard corpus of annotated data, any research work in this arena is bound to start from scratch, building up the training corpus by manual annotation. All the major previous research works in the field of citation analysis, related to the automatic classification of citation function have either generated their own data corpus by manual annotation of several citations [15], [1], [3] or have created rules by which the citations are labelled followed by careful scrutiny [12]. The Datasets used in all the previous works, for e.g. [12], [7] are NOT AVAILABLE due to either broken URL or since the researchers didn't make them available online.

But the major challenge encountered in this project is that since we define our Citation function label to be a pair of Sentiment polarity label and an Intent-Type Label, the dataset required in our work must be 'Multi-Labelled', i.e. It must consist of a pair of

{Sentiment Polarity Label, Citation Type Label} against each Citation Link

as single Training / Test Example.

After rigorous search, we conclude that such a data corpus has NEVER been created or collected since all the previous works are based on annotation corpora, either **ONLY the Citation Sentiment Polarity** or **ONLY the Type of Citing Intent**. Since Deep Neural Networks require a huge amount of training data sets at their disposal, we required some reliable source for generation of large datasets, for which a GUI-based utility tool was developed by the author to help in manual annotation of citations in adequate amounts, to be fed into any Deep Network- based architectures.

Apart from this, we also simultaneously worked on classification of citations according to their sentiment polarity using the dataset released online by [1].

Thus, initial work in the previous semester was primarily aimed at :

1. **Collection of Sufficient Datasets for feeding into the Neural Networks:** Our approach for this was to crawl papers from the Internet, followed by indexing them and then exploring a citation graph within the set of these papers and passing each example of the Citation Link to the Annotation tool interface.
2. **Classification of Citation Sentiment on the dataset available online:** We used SVM classifier to categorize citations based on their sentiment polarity, that gave moderate results.

The above part was the work done in the first phase (AUTUMN semester) of the project. In the second phase, we built architectures based on Neural Network framework to classify the same data corpus. Due to labour constraints and limited time, not much data could be annotated with the annotator tool utility built during the first phase and hence it was used just for testing the model.

For training the model, the dataset extracted from the URL http://cl.awaisathar.com/citation-sentiment-corpus/ was used since it contained sufficient annotated labels which is a very critical requirement for the purpose of training any given Deep Learning framework. However as mentioned earlier, this severely restricted us to work with ternary *{n,p,o}* classification labels as given in the dataset and we weren't able to test our architecture with Multi-labelled data as intended formerly.

2.1 Dataset (Citation Sentiment Classification)

The URL mentioned above was made available online by the researchers of [1] but it ONLY partially helps us as it contains just the sentiment polarity labels against each citation.

Nevertheless, since it satisfied our needs for a sufficiently large corpus, we proceeded by building a Neural Network framework for classification of citations, based on the Sentiment polarity of each citation as given in this dataset. So, the publicly available dataset from the URL

http://cl.awaisathar.com/citation-sentiment-corpus/ [1] was used on which we had earlier deployed an SVM-based classifier to classify the citations (in the first phase / previous semester). Now, keeping this SVM-based classifier as a baseline, we proceeded to building classifiers based on RNNs and Text-CNNs.

The data corpus consists of 8,736 citation sentences which have been manually annotated with sentiment. These citation sentences have been extracted from the ACL Anthology Network corpus. The features extracted and the results for this task have been further discussed in detail in the upcoming sections.

2.2 Dataset (Citation Function Classification)

As the need for a source of large datasets was felt, we decided to crawl research papers from the Net to collect sufficient citations and also developed a utility tool to facilitate easy manual annotation of citations so as to generate the required data corpus.

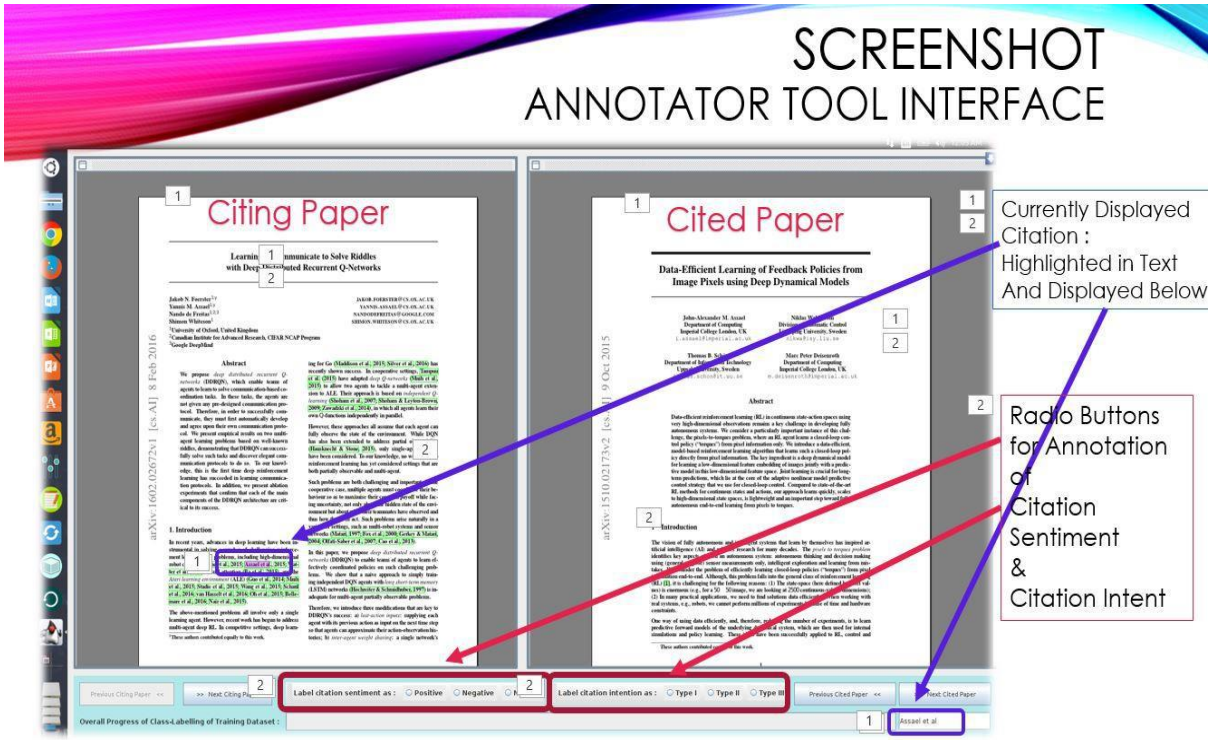


Fig 2.1 *Graphical User-friendly Interface for JAVA -based Annotator Utility Tool*

The Annotator utility tool has been developed in JAVA, and facilitates a nice Graphical User Interface, as shown in figure 2.1 to help quick annotation of citations. The approach of annotating the corpus follows in the sub-section,

supported by illustration of a self-explanatory block diagram as shown in figure 2.2.

2.2.1 Annotation Methodology

Since no publicly available dataset was found, we proceeded with generating our own annotated data corpus. Two online research repositories: ***arXiv e-print archive*** and ***Google Scholar*** were crawled to retrieve Citations and the JAVA-based Manual Annotator Utility Tool was deployed for generating the Dataset.

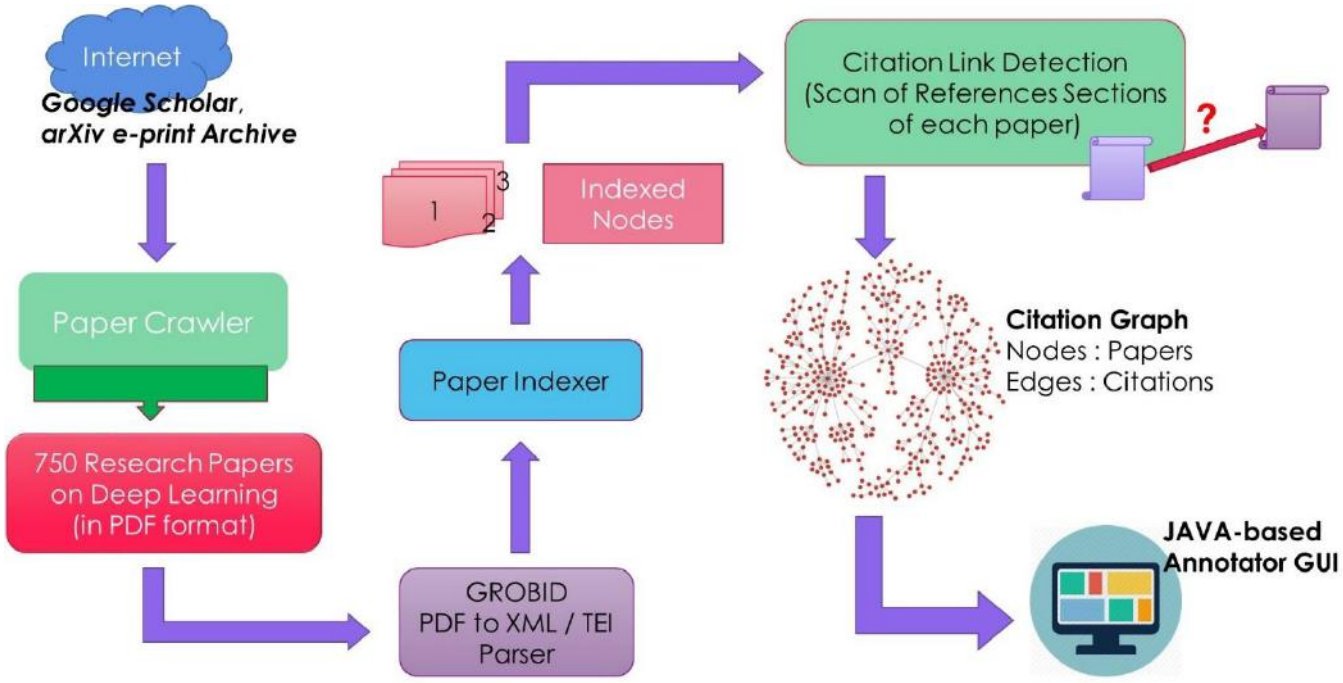


Fig 2.2 *Methodology Followed [Steps Taken (in form of Flowchart)]*

Crawling Research Papers

Selenium, a web scraper utility was used to build an automated crawler for retrieving sufficient research papers from the Web; the screenshot of the crawler in execution mode is shown in figure 2.3. The seed URL which was fed into the crawler was

http://arxiv.org:443/find/grp_cs/1/OR+OR+ti:+AND+deep+learning+ti:+AND+deep+neural+ti:+AND+deep+network/0/1/0/all/0/1

This was done so as to focus on a single research field, with the aim to find a dense citation network within the corpus. Since all the crawled papers have the terms “Deep Learning”, “Deep Neural” or “Deep Network” as their subsequence in their titles, all of them are works in the same field and are expected to cite many amongst each other, thus providing a dense citation graph for our annotation purposes.

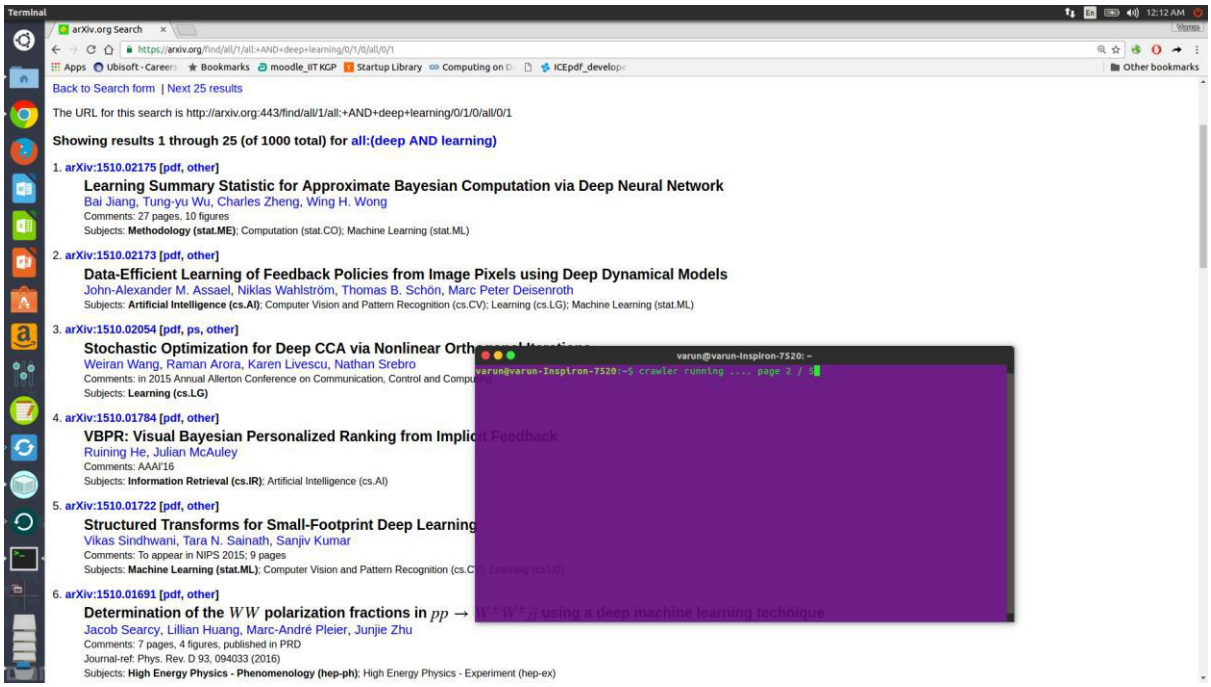


Fig 2.3 *Screenshot: Crawler running on e-print archive database arXiv*

Amongst all of the papers retrieved, we proceeded further with around 750 papers for the first attempt; the screenshot of the folder is shown in figure 2.4. Taking roughly at least 20 citations per node (paper), one would easily manage to collect around 15,000 annotated citations after manual labelling process. Such a figure, along with 8500 already available citations (with Sentiment labels) was expected to suffice for feeding into any architecture based on Deep Neural Networks.

The crawler has also been modified and extended to be able to retrieve papers from the Google Research Scholar and this would provide as an additional

source of citations, in case further amount of dataset needs to be acquired for Test/ Validation purposes.

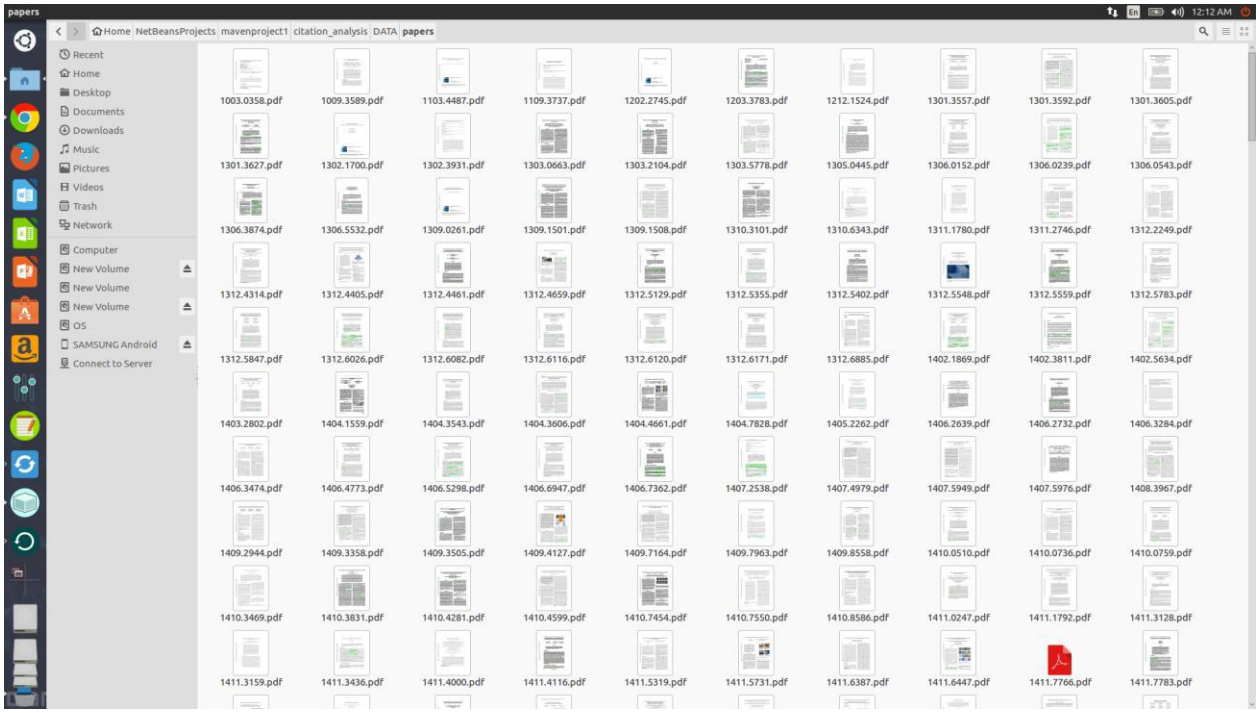


Fig 2.4 *Screenshot: 750 papers on deep learning collected for further processing*

Data Pre-processing:

PDF-Parsing, Paper Indexing & Exploration of Citation Graph

GROBID (***G***ene***R***ation ***O***f ***BI***bliographic ***D***ata) is a utility tool for parsing of PDF documents and extraction of bibliographic information and other contents from raw PDF files. It is a machine learning library for extracting, parsing and re-structuring raw documents such as PDF into structured TEI-encoded documents with a particular focus on technical and scientific publications.

This library was used to convert all the PDF documents into XML and all the citations of all the papers were extracted and indexed into a Hash Map. Each paper, thus was assigned a Node Index, thus acting as a vertex in the Citation Graph.

Next, the Citation Graph is explored in which all the bibliographic information extracted from papers i.e. the citations in the References section are scanned

through and matched against the nodes so as to obtain the Node IDs corresponding to the Citation Links.

Finally, we obtain a set of Edges / Links, along with the Node IDs of the Citing Paper as well as the Cited ones. This information is sent to the Annotator Utility Tool which displays each of the Citing-Cited pair on the interface, and saves the multi-label annotation it receives from the annotator. In order to simplify the task of the Human Annotator, an intelligent user-friendly interface (figure 2.5) also highlights the in-text citation reference sentence and the citing area (location in the citing paper) and facilitates easy feature extraction by offering suggestions and the distinguishing words (which help in classification) by finding similarity match against the previously annotated citations, thus enabling quick annotation of future citations.

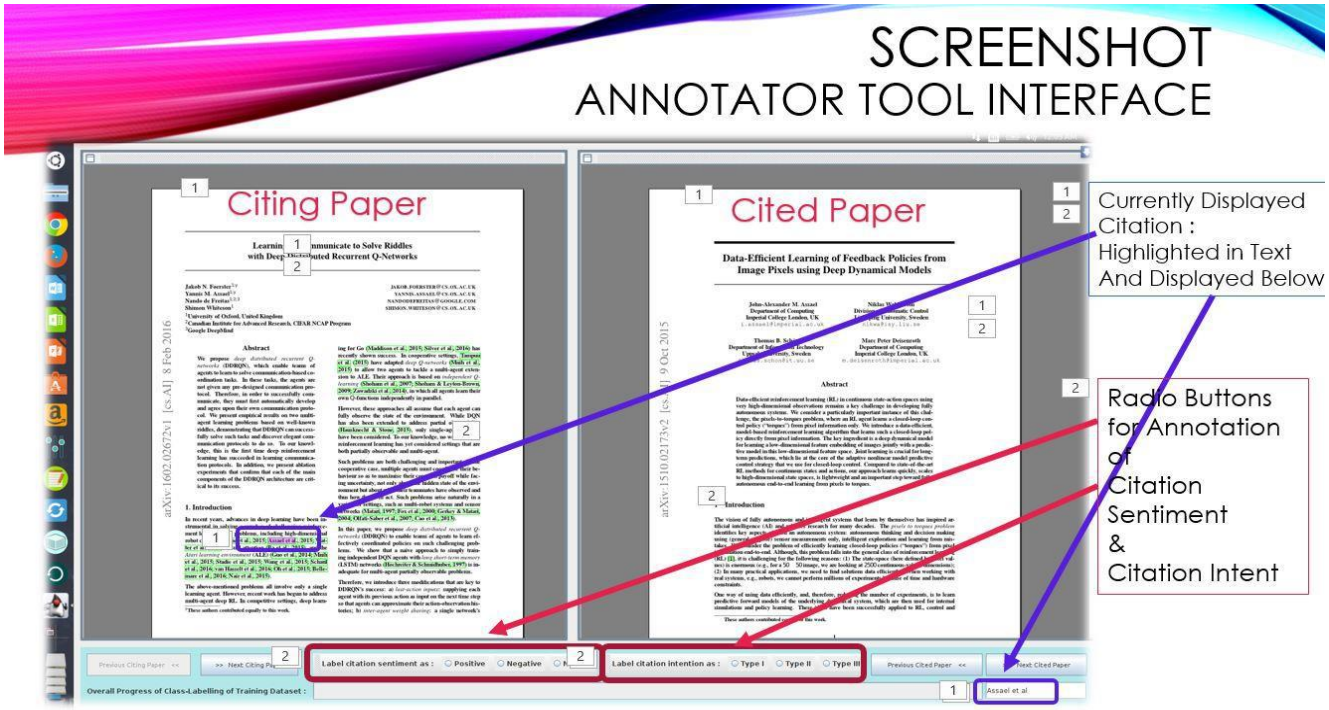


Fig 2.5 *Manual Annotation of Citation Network*

The annotated citations are accumulated into the corpus on the hard disk, which would serve as source of datasets for the Deep Neural Networks & and RNN Classifiers.

2.2.2 Self-Annotated Data Corpus

The annotator tool utility was used to generate as much labelled data as was possible in the given time-frame for pursuing the mentioned task, although, only 800 citation annotations could be developed till date. As a matter of fact, any Deep Learning-based framework cannot be trained on such a small-sized data corpus, it was eventually decided to choose the self-generated corpus (using the annotator-interface) only for testing phase whilst we trained the model with the publicly available corpus faceted only with 'citation sentiment', released by Awais [1] in the year 2011. Thus, we had to re-align our requirements accordingly and could not work with the full dual-label definition of the Citation Function that was proposed by us in section 1.2.1 of chapter I.

Since the training corpus was in conventional sentiment-only ternary label {n, p, o} format, we dropped the second part of the multi-class label (which reflects the citation intention type) while annotation of data using the developed tool.

The annotation was just a character label among N, P, O that meant the following:

'N' for Negative Sentiment: 244 citations,
'P' for Positive Sentiment: 743 citations, and
'O' for Neutral Sentiment: 6277 citations,

a snapshot of the same is shown in figure 2.6.

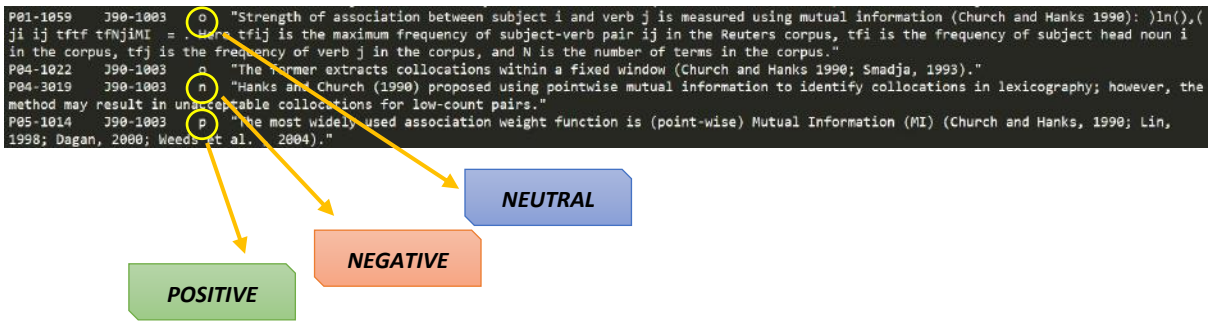


Fig 2.6 *Snapshots of Annotated Corpus*

2.3 Resolving Class Imbalance Problem

Similar to the first phase, we encountered one major problem during our experiments, that was the appearance of class-imbalanced datasets whose statistics are shown in figure 2.7.

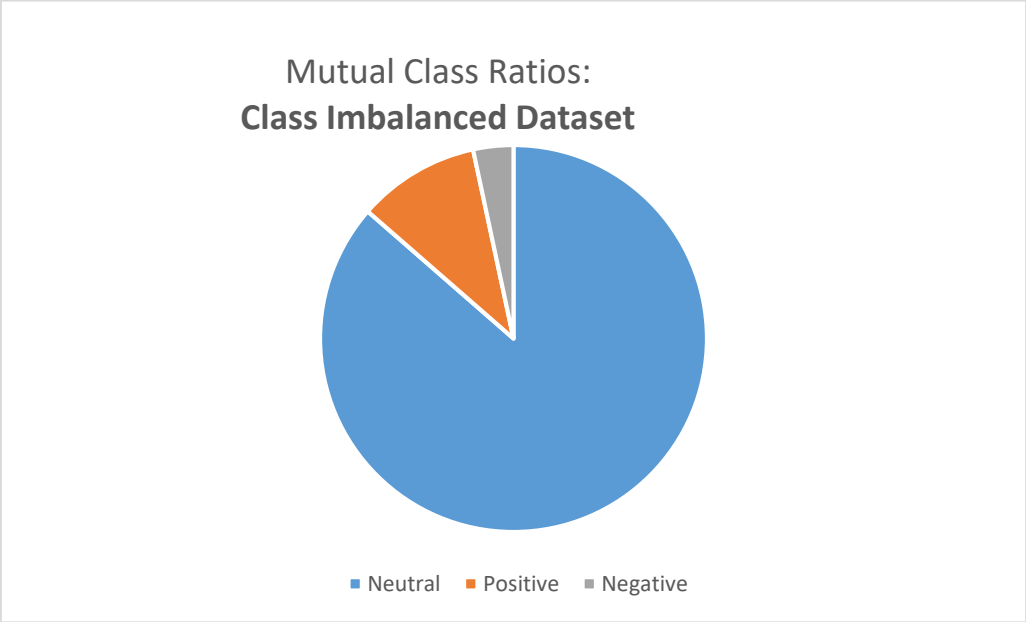
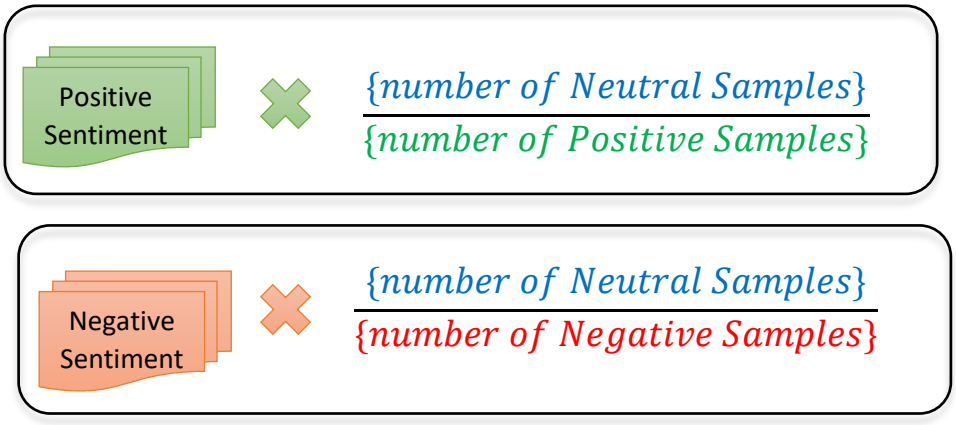


Fig. 2.7. Class Ratio Statistics of Imbalanced data sets

For this we used ***"Class-Balancing Techniques"*** such as:

(1) *Over-sampling* of the under-sampled classes (*Positive* & *Negative*) and
(2) *Under-sampling* of the over-sampled class (*Neutral*),

due to which the class ratio statistics changed as shown in figure 2.8.



Without such balancing of data, the results obtained would be superfluous and misleading as in any class-imbalanced scenario, where the trained model gets

biased to too much extent and learned to always output the majority proportion Class i.e. the *Neutral Sentiment Class* (class proportion: ~ 6000: 700: 200)

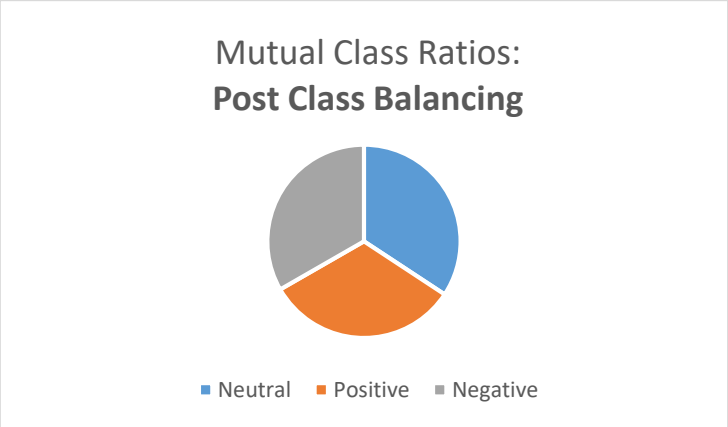


Fig. 2.8. Class Ratio Statistics after Class Balancing Step

Also, it is quite important to note that the results obtained without class – balancing were close to 95% which can easily be achieved by any constant function due to high proportion of the predicted class in the test dataset.

Thus, in such cases, accuracy is NOT at all a justified metric to evaluate the model performance on any grounds.

Chapter 3

Feature Extraction

The features selected for any Machine Learning task lies at the core of the whole process and play a very crucial role when comes to determining the accuracy and the overall performance of the classifier. Thus, the features have been carefully selected, after a very thorough analysis of all the previous works.

3.1 Features Extraction for Sentiment Classification

The features for this classification task have been borrowed from [1]. The goal was to replicate a LibSVM classifier as has been proposed in [1], and then to improve upon the results by deploying a RNN-based LSTM architecture for the same task. Thus, we have worked with the same features so as to establish a base for candid comparison.

The features extracted for classifying Citation sentiment can be broadly classified into three categories:

1. Word Level Features

As proposed in [1], we use unigrams and bigrams as features and also add 3-grams as new features to capture longer technical terms. POS tags are also included using two approaches:

1. attaching the tag to the word by a delimiter,
2. appending all tags at the end of the sentence.

This may help in distinguishing between homonyms with different POS tags and signalling the presence of adjectives respectively. Name of the primary author of the cited paper is also used as a feature. A science-specific sentiment lexicon is also added to the feature set. This lexicon consists of

83 polar phrases which have been extracted from the development set of 736 citations by [1]. Some of the most frequently occurring polar phrases in this set consists of adjectives such as efficient, popular, successful, state-of-the-art and effective.

2. ***Contextual Polarity Features***

   Since the task at hand is sentence-based, we use only the sentence-based features from the literature e.g., presence of subjectivity clues which have been compiled from several sources along with the number of adjectives, adverbs, pronouns, modals and cardinals. To handle negation, we include the count of negation phrases found within the citation sentence.

3. Sentence Structure Based Features

As suggested in [1], Dependency Structures and Sentence Splitting are the two different feature sets that have been exploited for the classification task. These features focus on the lexical and grammatical structure of a sentence and have not been explored previously for the task of sentiment analysis of scientific text.

Typed dependency structures describe the grammatical relationships between words and tend to capture the long-distance relationships between words. For instance, in the sentence "… *showed that the results for French-English were competitive to state-of-the-art alignment systems*", the relationship between results and competitive will be missed by trigrams but the dependency representation captures it in a single feature: ***nsubj_competitive_results.*** In case of sentence splitting, the objective is to remove irrelevant polar phrases around a citation that is expected to improve results, for which we split each sentence by trimming its parse tree.

3.2 Feature Extraction in Annotator Utility Tool

The overall approach for feature extraction has been borrowed from the works in [12] and [3].

3.2.1 Identification of Citing Area

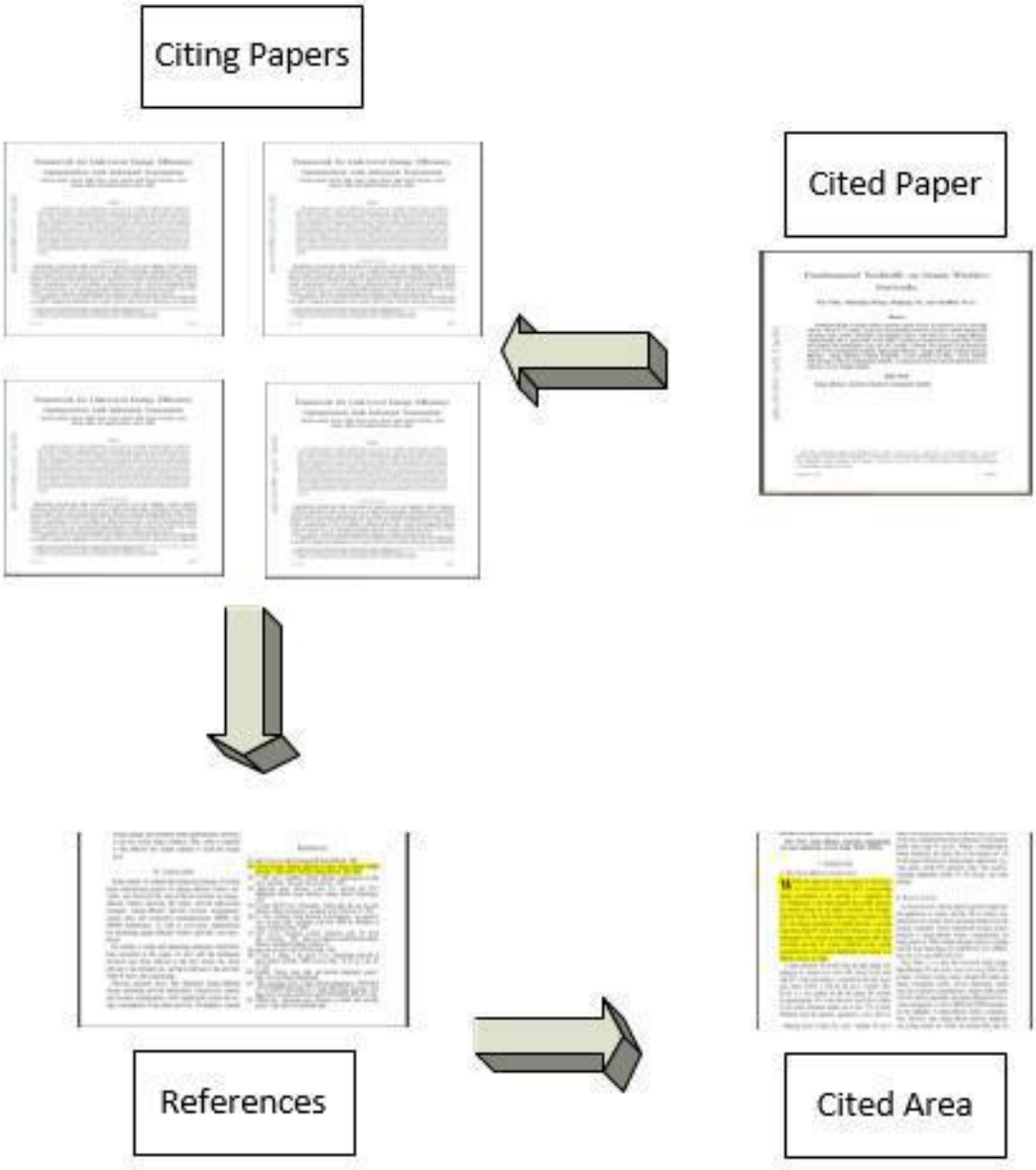


Fig 3.1 *Identification of Citing Area*

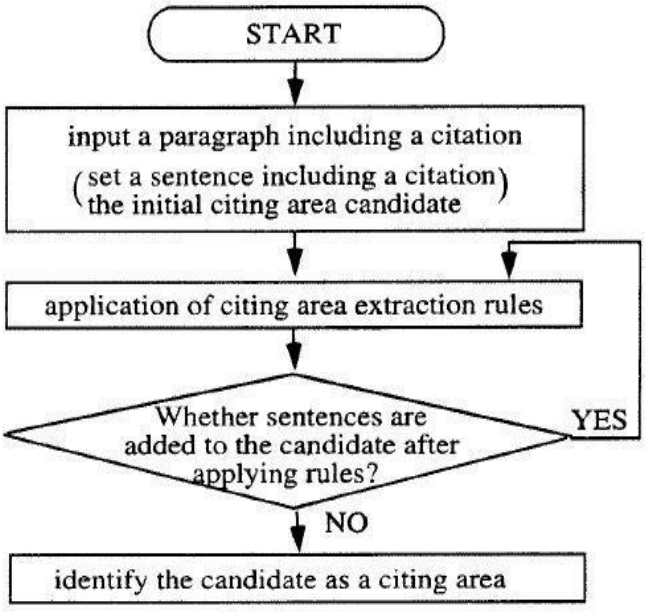


Fig 3.2 *Identification of Citing Area*

First, we need to identify the "Citing Area", which can range from a single citation sentence to a few small paragraphs. This is essential because the scope of influence of the citation can extend up to a few lines above or below the actual reference / citation. This is an important preliminary step before feature extraction, so as to guarantee the completeness of the process of extraction of all the relevant features in the nearby text area.

As depicted in Fig 3.2, the task of identification of the most relevant set of sentences, which together make up the "Citing Area", can be accomplished by the method suggested in [12]:

Initially, a citing area candidate consists of a single sentence that contains the citation in consideration. The citing area extraction rules (i.e. the presence of any one of the pre-defined list of polarity words, cue phrases, etc.) are applied to the sentence just before and just after the citing area candidate determined so far (always staying within the selected paragraph). If either sentence includes any of the cue phrases (fig 3.3), it is added to the citing area candidate. This process continues until no more sentences are added.

types	cue phrases
(1) anaphor	For this, For these, On this, On these, In this, in this in these, In these, This, These, Therefore
(2) negative expression	yet, less, but, in spite of, unlike, rarely in contrast, although, Still, Nevertheless, instead, despite, irrelevant, has not been, not attempt to not possible to, this is not, but is not, less, has not, have not
(3) 1st person pronoun	I, in our example, our analysis was, our analysis of by using our, in our work, our analysis is, to our concept, our analysis, our work our example, using our, we
(4) 3rd person pronoun	they, their, them, he, his, him, she her, hers
(5) adverb	And, Furthermore, Because, Again, Additionally, Such, In such, So
(6) other	difference between, defect, drawback, impossible, Using, we incorporate, in the implementation, is implemented, first, second, theory, theoretical, origin, based, base, basis, adopt, apply, applied, foundation, fundamental, radical, element, underlie, underlay, underlain, underlying, In particular, follow

Fig 3.3 Cue phrases for i*dentification of Citing Area and extraction of features*

3.2.2 Extraction of Features from Citing Area

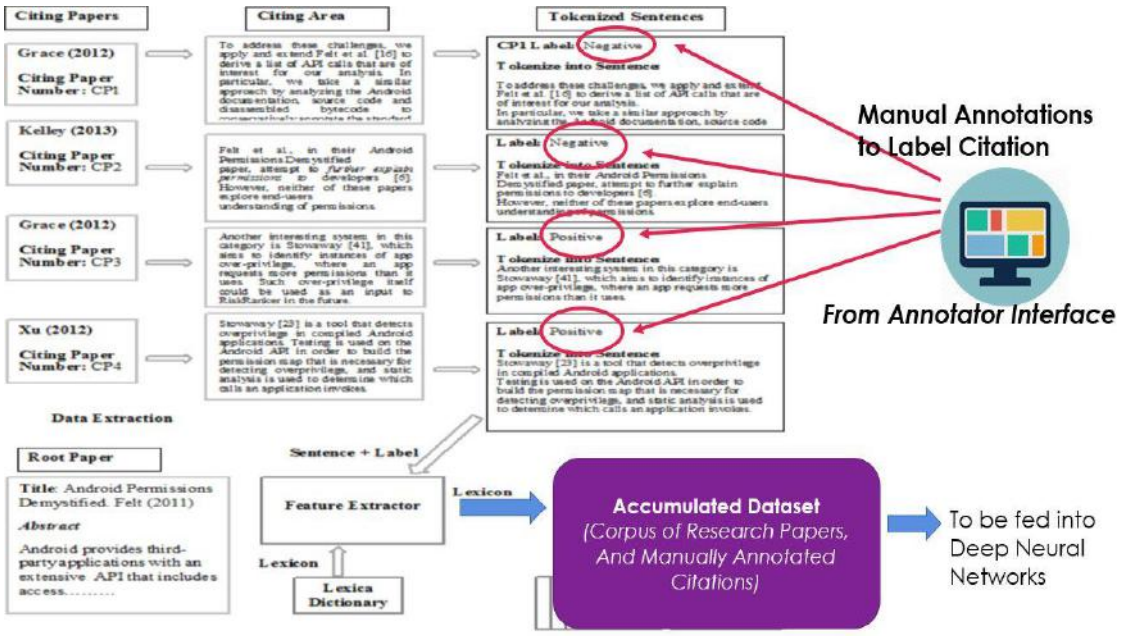


Fig 3.4 Feature Extraction Process

Positive and Negative lexica obtained from Evert (2014) consisting of approximately 28,000 distinct positive and 31,000 distinct negative words as well as Positive Incrementor and Negative Incrementor derived from [12], each of which includes approximately 75 entries that are unigrams, bigrams or trigrams, were used.

Feature Extractor iteratively reads each of the sentences in the citation area. Firstly, the citing area is tokenized into sentences to compare with Positive and Negative Incrementor, and then each sentence is tokenized into words to compare with Positive and Negative Dictionaries. For each match and mismatch between token and dictionary we create a feature-set list, composed of Token ('1' for match and '0' for mismatch) and Label of that citing paper.

Other than pre-fixed set of cue phrases, we also brought to use a technique in which the human annotator is asked to select the meta-description cue phrases that justify their choice of category, during the annotation phase.
This is done by just fragmenting each of the sentences in the citation area into separate words that are presented in the form of Toggle Buttons which can be selected or de-selected. Some pre-fed cue phrases if found by the automated tool, will be provided pre-selected.

All these features are embedded along with the Citation Link corresponding to the given Citation, and saved into the Database.

Since Manual Annotation takes time, we expect the process of annotation to complete by December, 2016 after which we the project will proceed to the deployment of Deep Neural Network- based frameworks.

Until then, we carried out Feature Extraction and Classification tasks using already available Sentiment Classification dataset.

3.3 Feature Vector Inputs

Each architecture was trained and tested on two different types of feature-level data:

3.3.1 Word Vector Input

The RNN and the CNN architectures were built and the models based on these architectures were fed raw sentence data (of course after a little pre-processing). Each of the citation sentence was first concatenated to a large text corpus, which after stop word removal, lemmatization, stemming and tokenization (general NLP pre-processing) was saved as an indexed vocabulary. Then, the word vectors corresponding to these words were directly fed into the LSTM or CNN one by one, per sentence, along with their target sentiment label.

This was also tried in two different ways:

1. Fixing the word vectors to those pre-trained by unsupervised learning (Word2Vec Word Embeddings) and disabling their training

2. Random initialization of word embedding layer and enabling simultaneous training of the word vectors with optimization techniques so as to keep them task-specific.

Out of the two ways, it was observed that the second alternative gave comparatively better results than the first and this can very well be justified and attributed to the fact that the task at hand i.e. of classification of the citation sentiment is quite sensitive to the context-relative words which cannot be fulfilled by generic unsupervised learning.

3.3.2 Feature-equipped Word Vector Input

Another alternative was to use the features extracted as elaborately discussed in the Section 3.1 by directly placing them along with the word vectors. That is, the now effective word vectors that were fed into the architectures were the word embedding of the words of the citation sentence concatenated with the feature descriptors (of the features described in Section 3.1) extracted from that respective sentence as shown in the figure below.

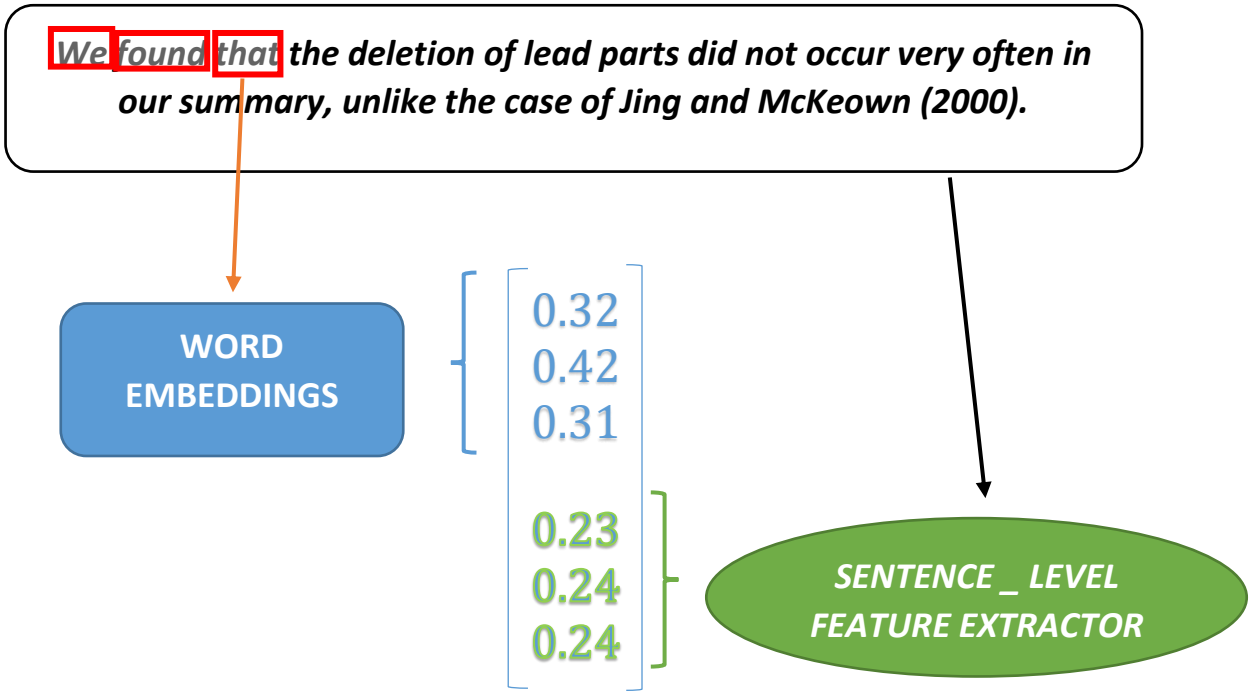


Fig 3.5 Feature Extracted Word Vectors – Concatenation of Word Embeddings with Sentence-Level Feature Descriptors

It was observed in the experiments that this alternative outperformed the others and is also a more valid way of comparison with the baseline SVM model since it provides the Neural Network architecture with the same features as was available to the SVM Model.

Chapter 4

Citation Sentiment Classification - Baseline SVM Classifier

In the minor phase of the project tenure, Lib SVM Classifier from the WEKA Machine Learning Toolkit was deployed for the Citation Sentiment Classification task. The data corpus containing 8736 manually annotated citations was suitably split into train and test partitions, by the WEKA classification tools.

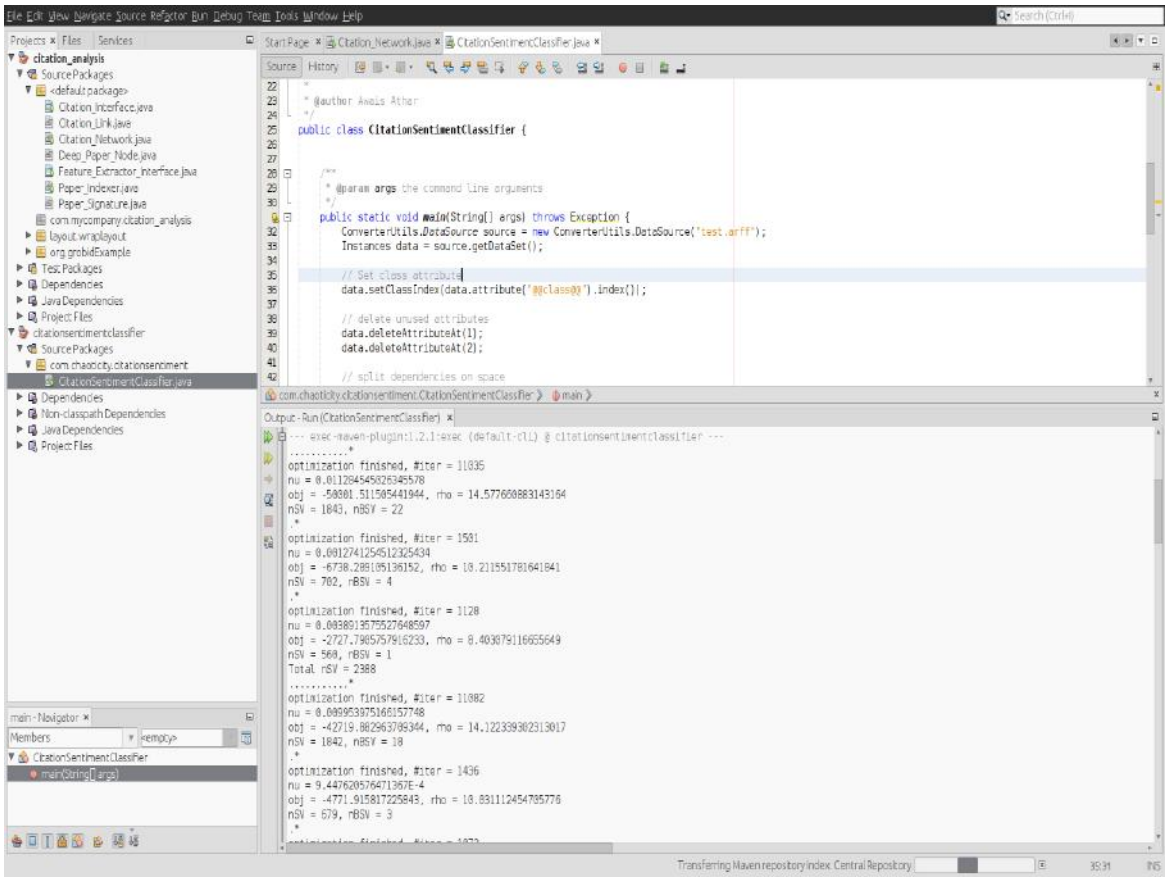


Fig 4.1 *LibSVM @ Weka Toolkit running iterations on Sentiment Classification Task*

The classification results achieved after the AUTUMN phase of the project execution were found moderately good and it was expected then that the results would improve upon deployment of RNN-based architectures, LSTMs to be implemented during SPRING phase of project tenure. A screenshot of classification results can be seen in the figures 4.2 and 4.3 below:

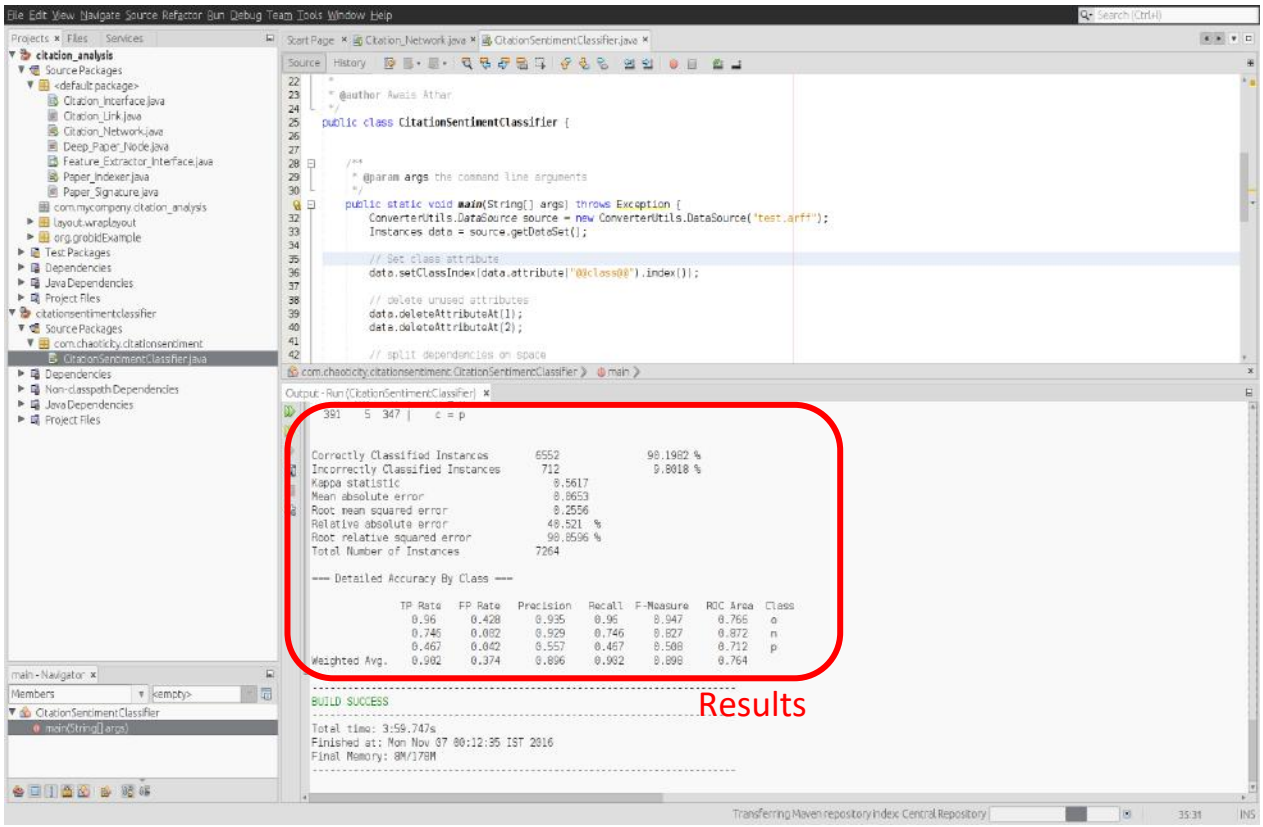


Fig 4.2 *Citation Sentiment Classification Results with Lib SVM Classifier*

Confusion Matrix

Classified As / Actual	Positive	Negative	Neutral
Positive	347	5	391
Negative	31	182	31
Neutral	245	9	6023

Summary

• Correctly Classified Instances	6552	(90.1982 %)
• Incorrectly Classified Instances	712	(9.8018 %)
• Kappa statistic	0.5617	
• Mean absolute error	0.0653	
• Root mean squared error	0.2556	
• Relative absolute error	40.521 %	
• Root relative squared error	90.0596 %	
• Total Number of Instances	7264	

Detailed Accuracy By Class

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
0.96	0.428	0.935	0.96	0.947	0.766	Neutral
0.746	0.002	0.929	0.746	0.827	0.872	Negative
0.467	0.042	0.557	0.467	0.508	0.712	Positive
0.902	0.374	0.896	0.902	0.898	0.764	Weighted Avg.

Fig 4.3 *Citation Sentiment Classification Result Summary Statistics Using Lib SVM Classifier*

Chapter 5

Neural Network Architectures for Citation Sentiment Classification

We have deployed two major architectures which are quite popular in solving tasks in their respective areas:

1. ***RNN-based LSTMs*** and
2. ***CNNs.***

The general diagram demonstrating the use of these architectures for the task of sentiment classification has been shown in the figure.

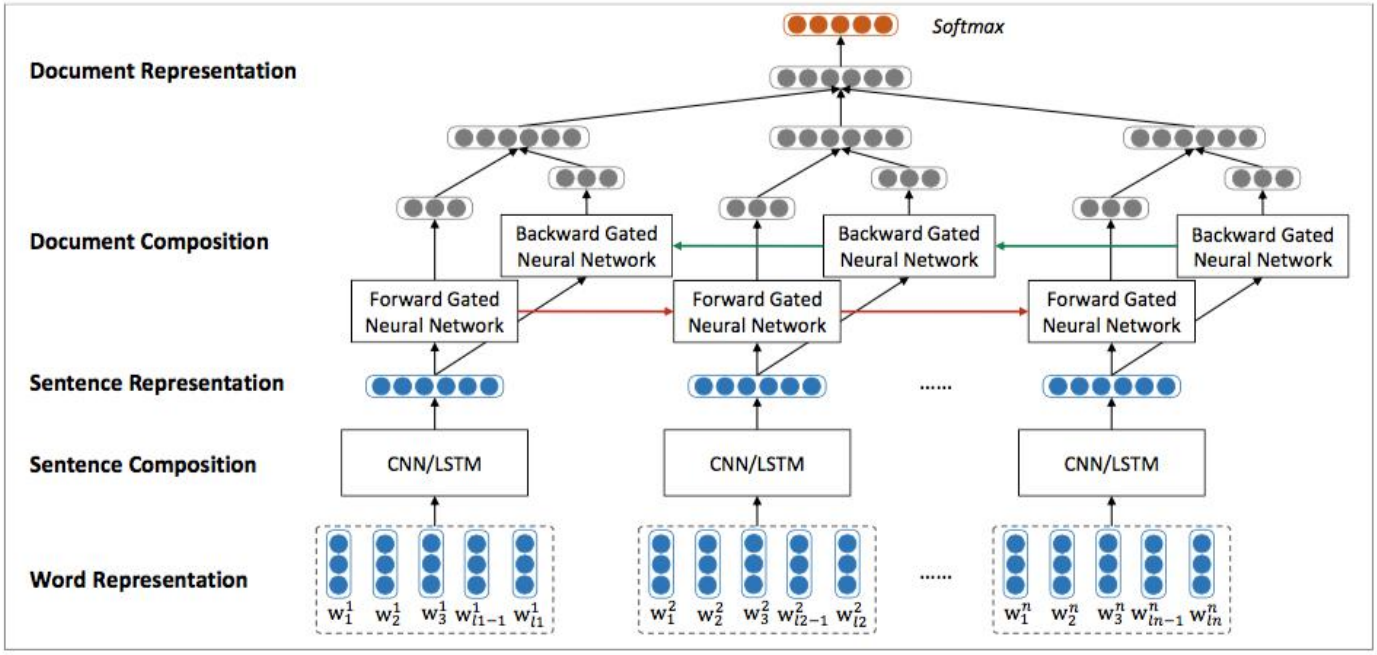


Fig 5.1 *CNN / LSTM architecture for Sentiment Classification task*

Since there have been developed a multitude of several different variations of such architectures, we further discuss on the exact architecture variants we implemented for the task in section 5.1.3. Prior to that, we also briefly shed some light on a broad idea of their general structure, construction and design for the uninformed readers, and then distinctly clarify the details of the specific variant we used in our project.

5.1 Deep Bidirectional LSTMs

5.1.1 Introduction to RNNs, LSTMs

Recurrent Neural Networks (RNNs) are popular models that have shown great promise in many NLP-related tasks.

The idea behind RNNs is to make use of sequential information. In a traditional neural network, it is assumed that all inputs (and outputs) are independent of each other. But for many tasks that's not at all an appropriate idea. If one wants to predict the next word in a sentence he better knows, which words came before it. RNNs are called *recurrent* because they perform the same task for every element of a sequence, with the output being depended on the previous computations. Another way to think about RNNs is that they have a "memory" which captures information about what has been calculated so far. In theory RNNs can make use of information in arbitrarily long sequences, but in practice they are limited to looking back only a few steps.

A typical RNN looks as follows:

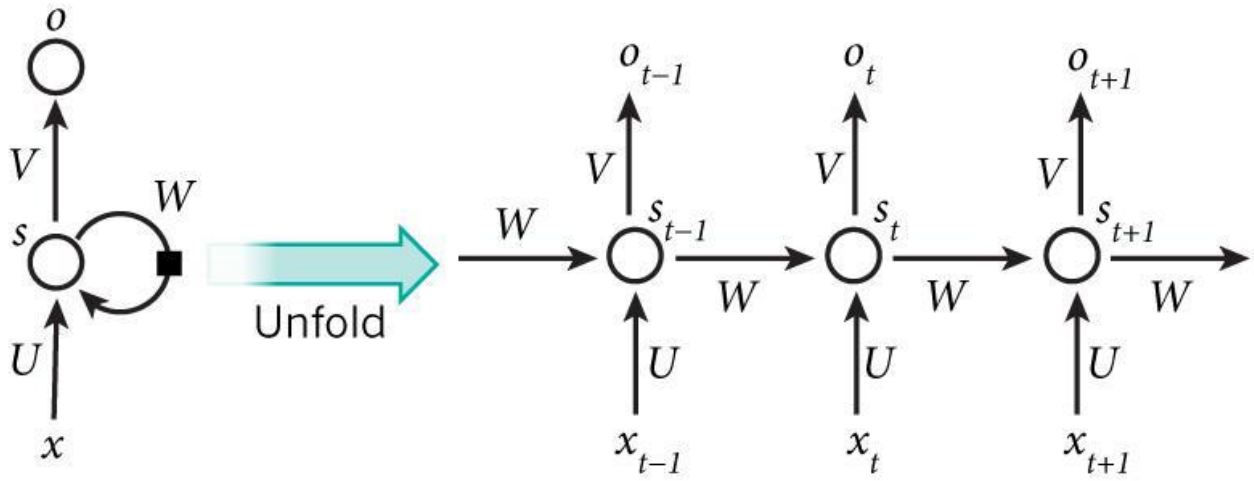


Fig 5.1 *A recurrent neural network and the unfolding in time of the computation involved in its forward computation*

The above figure 5.1 shows a RNN being *unrolled* (or unfolded) into a full network. By unrolling it is implied that one writes out the network for the complete sequence. For example, if the sequence we care about is a sentence of 5 words, the network would be unrolled into a 5-layer neural network, one

layer for each word. The formulas that govern the computation happening in a RNN are as follows:

- x_t is the input at time step t. For example, x_1 could be a one-hot vector corresponding to the second word of a sentence.
- s_t is the hidden state at time step t. It's the "memory" of the network. s_t is calculated based on the previous hidden state and the input at the current step: $s_t = f(Ux_t + Ws_{t-1})$. The function f usually is a nonlinearity such as tanh or ReLU. s_{-1}, which is required to calculate the first hidden state, is typically initialized to all zeroes.
- o_t is the output at step t. For example, if we wanted to predict the next word in a sentence it would be a vector of probabilities across our vocabulary. $o_t = \mathrm{softmax}(Vs_t)$.
- The hidden state s_t can be thought of as the memory of the network. s_t captures information about what happened in all the previous time steps. The output at step o_t is calculated solely based on the memory at time t. In practice, it's a bit more complicated simply because s_t typically can't capture information from too many time steps ago.

PARAMETER SHARING:

Unlike a traditional deep neural network, which uses different parameters at each layer, a RNN shares the same parameters (U, V, W above) across all steps. This reflects the fact that we are performing the same task at each step, just with different inputs. This greatly reduces the total number of parameters the model needs to learn.

The above figure 5.1 has outputs at each time step, but depending on the task this may not be necessary. When predicting the sentiment of a sentence we may only care about the final output, not the sentiment after each word. Similarly, we may not need inputs at each time step. The main feature of an RNN is its hidden state, which captures some information about a sequence.

In practice, such Simple RNNs are severely limited in capacity to model real-life texts, due to their insufficient capability to look back only a few steps. Thus, keeping such serious limitations into consideration, especially for long-term-dependency-critical tasks like sentiment classification, a variant of RNN-based architectures designed especially for solving long-term dependencies, called the

Long Short-Term Memory Networks (LSTMs) has been shown to outperform almost every other RNN-based model. LSTMs perform essentially the same process as the RNNs, they just have a different way of computing the hidden state which allows the underlying architecture to better represent the patterns found in the text. Technically, they possess additional gates to enable the architectures to explicitly choose the information to be forgotten or stored in the newly updated state which consequently leads to an increase in its expressive power and capability.

Long Short Term Memory networks – usually just called "LSTMs" – are a special kind of RNN, capable of learning long-term dependencies. They were introduced by [6] and were refined and popularized by many people in following work. They work tremendously well on a large variety of problems, and are now widely used. LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically their default behaviour, not something they struggle to accomplish.

All recurrent neural networks have the form of a chain of repeating modules of neural network. In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer; the block diagram can be seen in figure 5.2.

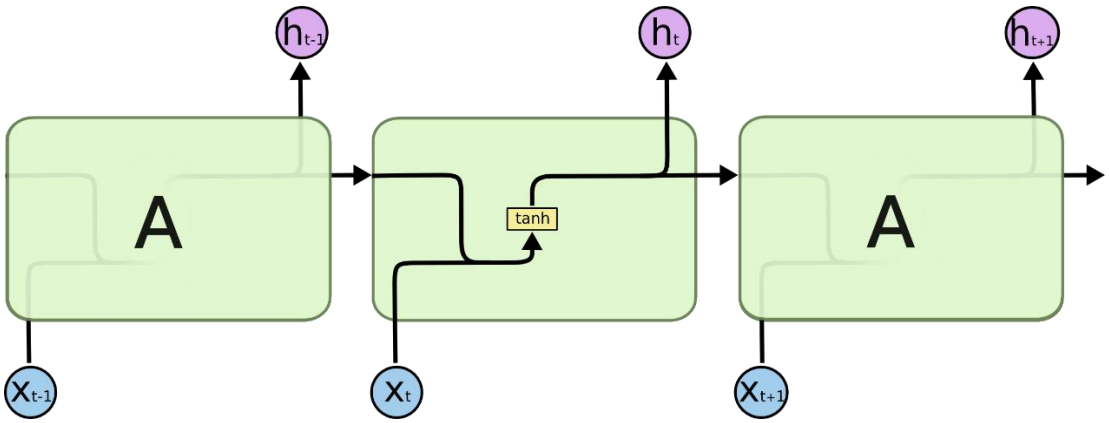


Fig 5.2 *The repeating module in a standard RNN contains a single layer.*

As can be seen in figure 5.3, LSTMs also have this chain like structure, but the repeating module has a different structure. Instead of having a single neural network layer, there are four, interacting in a very special way.

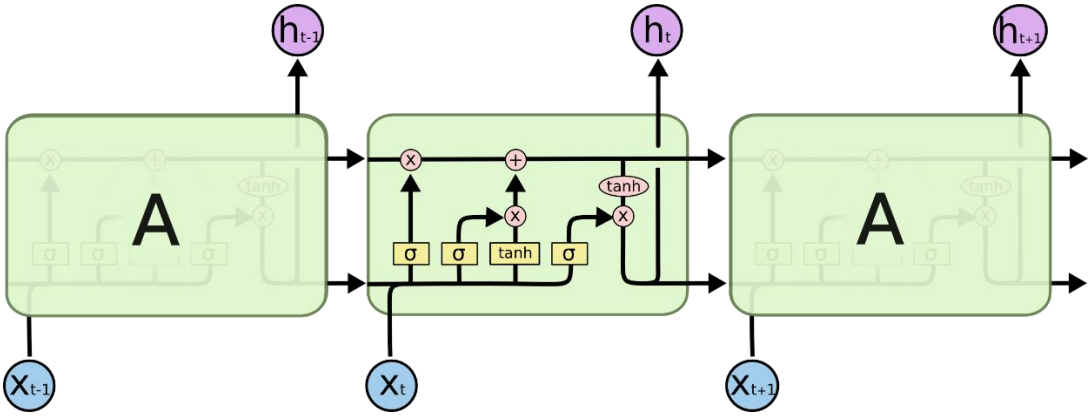
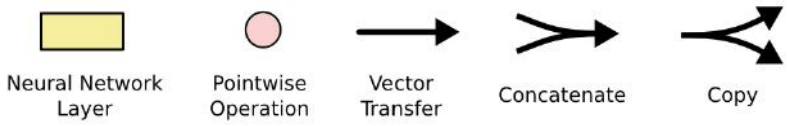
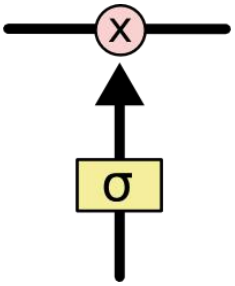


Fig 5.3 *The repeating module in an LSTM contains four interacting layers.*



The key to LSTMs is the cell state, the horizontal line running through the top of the diagram, which is kind of like a conveyor belt. It runs straight down the entire chain, with only some minor linear interactions. It's very easy for information to just flow along it unchanged.



The LSTM does have the ability to remove or add information to the cell state, carefully regulated by structures called gates, which provide a way to optionally let information through. They are composed out of a pointwise multiplication operation and a sigmoid neural net layer which outputs numbers between zero and one, describing how much of each component should be let through. A value of zero means "let nothing through," while a value of one means "let everything through!". An LSTM has three of these gates, to protect and control the cell state.

5.1.2 Bidirectional LSTMs

Bidirectional RNNs are based on the idea that the output at a given time may not only depend on the previous elements in the sequence, but also future

elements. They are implemented in the form of two RNNs stacked on top of each other. The output is then computed based on the hidden state of both RNNs**.** ***Deep bidirectional RNNs*** are similar to bidirectional RNNs, only that we now have multiple layers per time step, a schematic block diagram can be seen in figure 5.4.

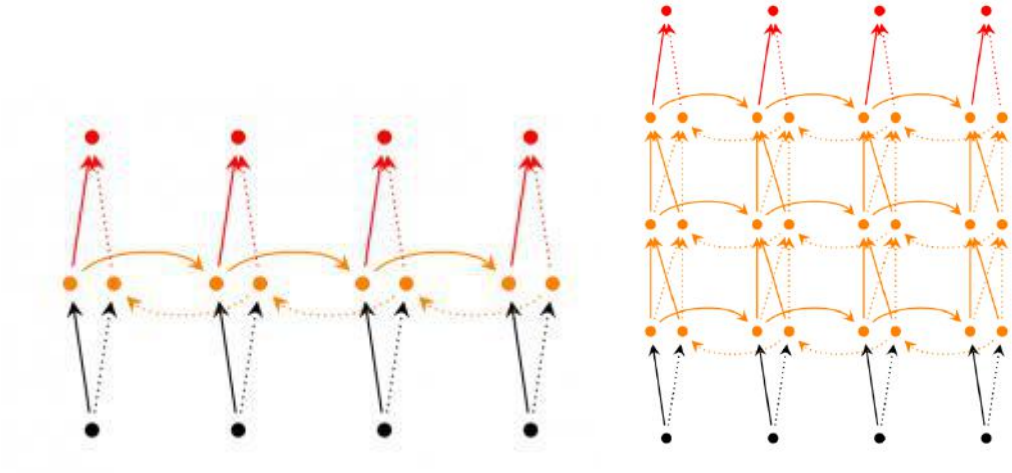


Fig 5.4 *Bidirectional RNNs & Deep (Multi-Layered Bidirectional RNNs)*

Although we implemented several different variants of ***bidirectional LSTMs*** on our dataset, we observe that there are only negligible variations in the final results we obtain with them and that is still the case even on varying the hyperparameter values and hence we report the results on only one of them.

5.1.3 Implemented LSTM Model

The model we implemented is a **simple Multi-Layered ("*Deep*") Bidirectional LSTM Neural Network,** which we explained broadly in the previous sections.

The model can be visualized in the following figure 5.5, where the hidden state averaged across all time steps is what is passed to the logistic regression layer.

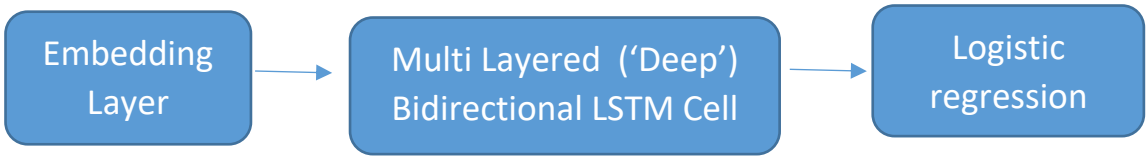


Fig 5.5 *Our Architecture - FLOWCHART*

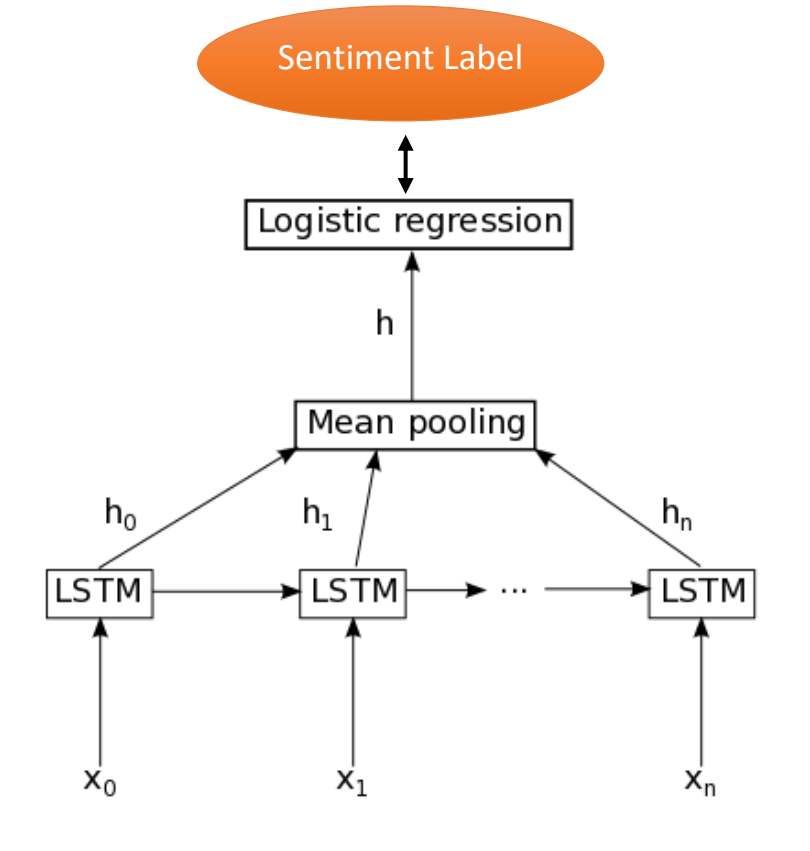


Fig 5.6 *Implemented LSTM Model*

5.2 Text CNNs

A ***Convolutional neural network*** (**CNN**, or **ConvNet**) is a type of feed-forward artificial neural network in which the connectivity pattern between its neurons is inspired by the organization of the animal visual cortex. Individual cortical neurons respond to stimuli in a restricted region of space known as the receptive field. The receptive fields of different neurons partially overlap such that they tile the visual field. The response of an individual neuron to stimuli within its receptive field can be approximated mathematically by a convolution operation.

5.2.1 Introduction to CNNs

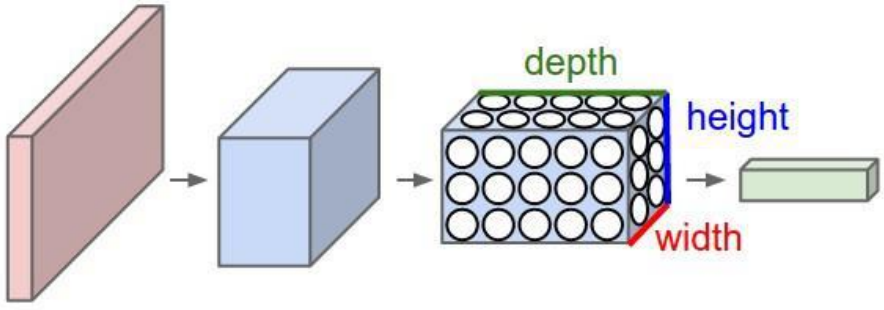


Fig 5.7 *Convolutional Neural Network*

A ConvNet differs from regular neural networks in terms of the flow of signals between neurons. Typical neural networks pass signals along the input-output channel in a single direction, without allowing signals to loop back into the network. This is called a forward feed. A CNN can be described in four main steps: Convolution, Subsampling, Activation and full connectedness. The most popular implementation of the CNN has been proposed in LeNet in [9]. The operational layers of a convolutional neural network consist of several layers which are broadly of three types:

- **Convolutional layers** consist of a rectangular grid of neurons. It requires that the previous layer *also* be a rectangular grid of neurons. Each neuron takes inputs from a rectangular section of the previous layer; the weights for this rectangular section are the same for each neuron in the convolutional layer. Thus, the convolutional layer is just an image *convolution* of the previous layer, where the weights specify the convolution filter.
- **Max-Pooling:** After each convolutional layer, there may be a pooling layer. The pooling layer takes small rectangular blocks from the convolutional layer and subsamples it to produce a single output from that block. Our pooling layers will always be max-pooling layers; that is, they take the maximum of the block they are pooling.
- **Fully-Connected:** Finally, after several convolutional and max pooling layers, the high-level reasoning in the neural network is done via fully connected layers. A fully connected layer takes all neurons in the previous layer (be it fully connected, pooling, or convolutional) and connects it to every single neuron it has.

5.2.2 Text CNNs for NLP Tasks

In [8], Yoon Kim first introduced the concept of deploying ConvNets for solving supervised machine learning tasks related to Natural Language Processing.

It was observed to outperform several other frameworks for NLP classification tasks. We implemented such a Text CNN Model for our classification task of Citation Sentiment.

In the project work, we trained a simple CNN with one layer of convolution over the word vectors which were obtained in two alternative ways :

(1) Pre-trained using the unsupervised neural language model of Word2Vec [10]
(2) Simultaneously training the word vectors to keep them task-specific

The first alternative is based on the assumption that the pre-trained vectors are 'universal' feature extractors that can be utilized for various classification tasks. Learning task-specific vectors in the second alternative results in further improvements. The model architecture [8] is as shown in the figure :

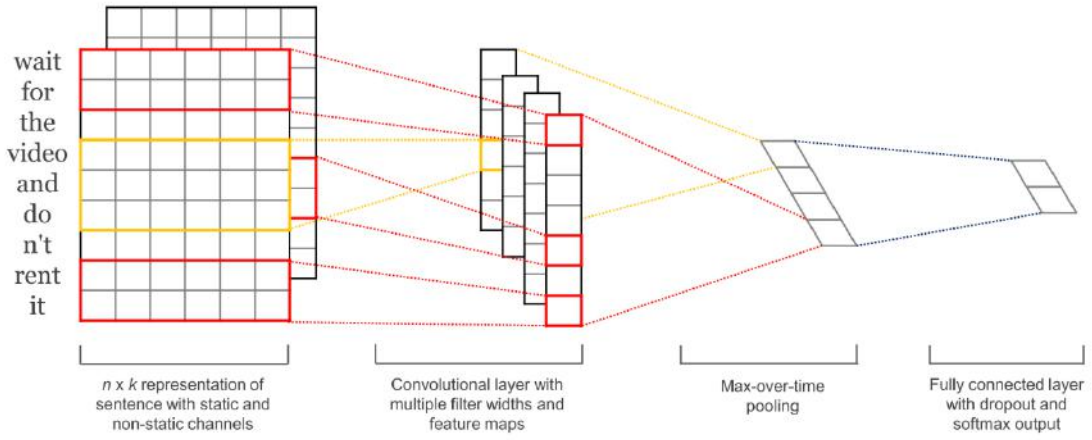


Fig 5.8 *CNN – based architecture for Sentiment Classification Task*

5.2.3 Implemented Text-CNN Model

The model architecture, shown in figure above, is a slight variant of the CNN architecture of [4]. Let $x_i \in \mathbb{R}^k$ be the k -dimensional word vector corresponding to the i-th word in the sentence. A sentence of length n (padded wherever necessary) is represented as

$$X_{1:n} = X_1 \oplus X_2 \oplus \ldots \oplus X_n, \qquad (1)$$

where $\oplus$ is the concatenation operator. In general, let $x_{i:i+j}$ refer to the concatenation of words $x_i, x_{i+1}, \ldots\ldots, x_{i+j}$. A convolution operation involves a *filter* $\boldsymbol{w} \in \mathbb{R}^{hk}$, which is applied to a window of h words to produce a new feature. For example, a feature c_i is generated from a window of words $x_{i:i+h-1}$ by

$$c_i = f\,(\, w.X_{i:i+h-1} + b\,) \tag{2}$$

Here b $\in \mathbb{R}$ is a bias term and f is a non-linear function such as the hyperbolic tangent. This filter is applied to each possible window of words in the sentence $\{\, X_{1:h}, X_{2:h+1}, \ldots\ldots, X_{n-h+1:n}\,\}$ to produce a *feature map*:

$$\boldsymbol{c} = \,[\, c_{1:h}, c_{2:h+1}, \ldots\ldots, c_{n-h+1:n}\,] \tag{3}$$

With $c \in \mathbb{R}^{n-h+1}$. We then apply a max-overtime pooling operation [4] over the feature map and take the maximum value $\hat{c} = \max\{\boldsymbol{c}\}$ as the feature corresponding to this particular filter. The idea is to capture the most important feature—one with the highest value—for each feature map. This pooling scheme naturally deals with variable sentence lengths.

The process describes the way '*one*' feature is extracted from '*one*' filter. The model uses multiple filters (with varying window sizes) to obtain multiple features. These features form the penultimate layer and are passed to a fully connected SoftMax layer whose output is the probability distribution over labels.

In one of the model variants, we experiment with having two 'channels' of word vectors—one that is kept static throughout training and one that is fine-tuned via backpropagation as shown in figure 5.9. In the multichannel architecture, illustrated in the figure, each filter is applied to both channels and the results are added to calculate c_i in equation (2). The model is otherwise equivalent to the single channel architecture. For regularization, we employ dropout on the penultimate layer with a constraint on l2-norms of the weight vectors [5].

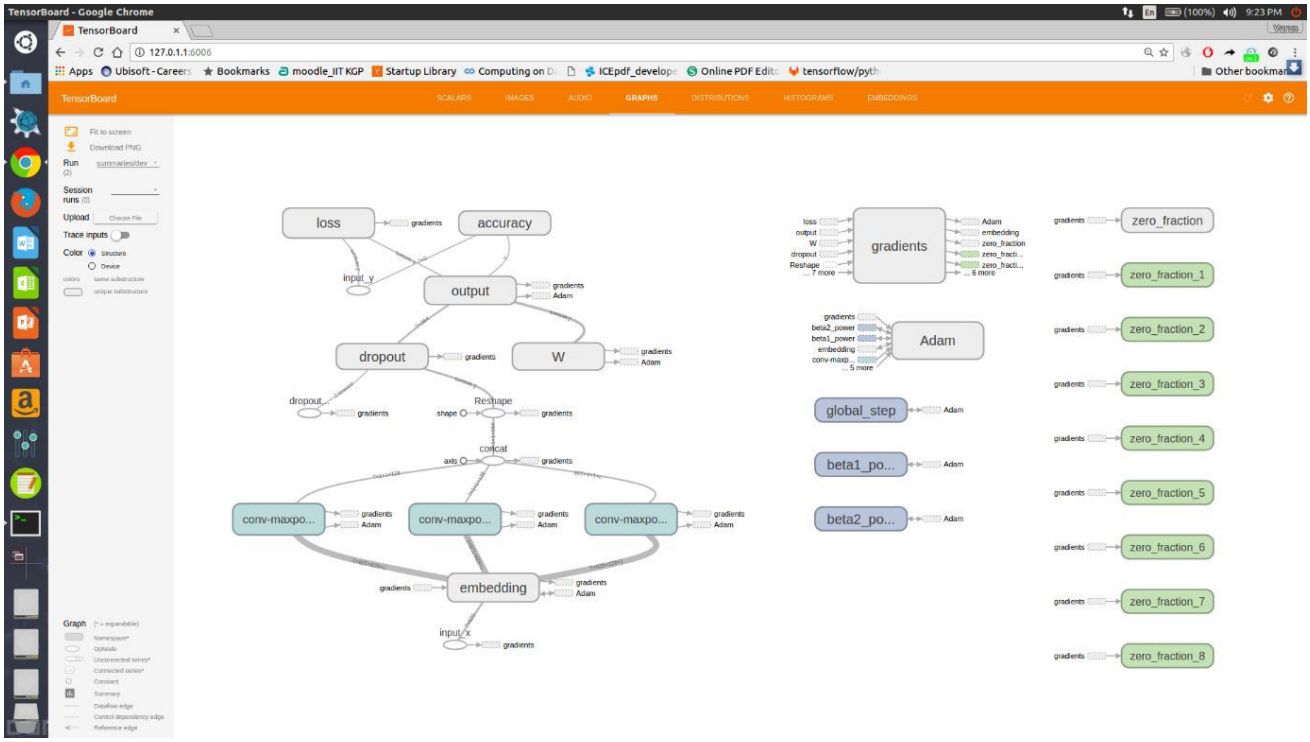


Fig. 5.9 *Text CNN Model Architecture Visualization in TENSOR-BOARD*

Chapter 6

Citation Sentiment Classification Results Using Deep Neural Network Models

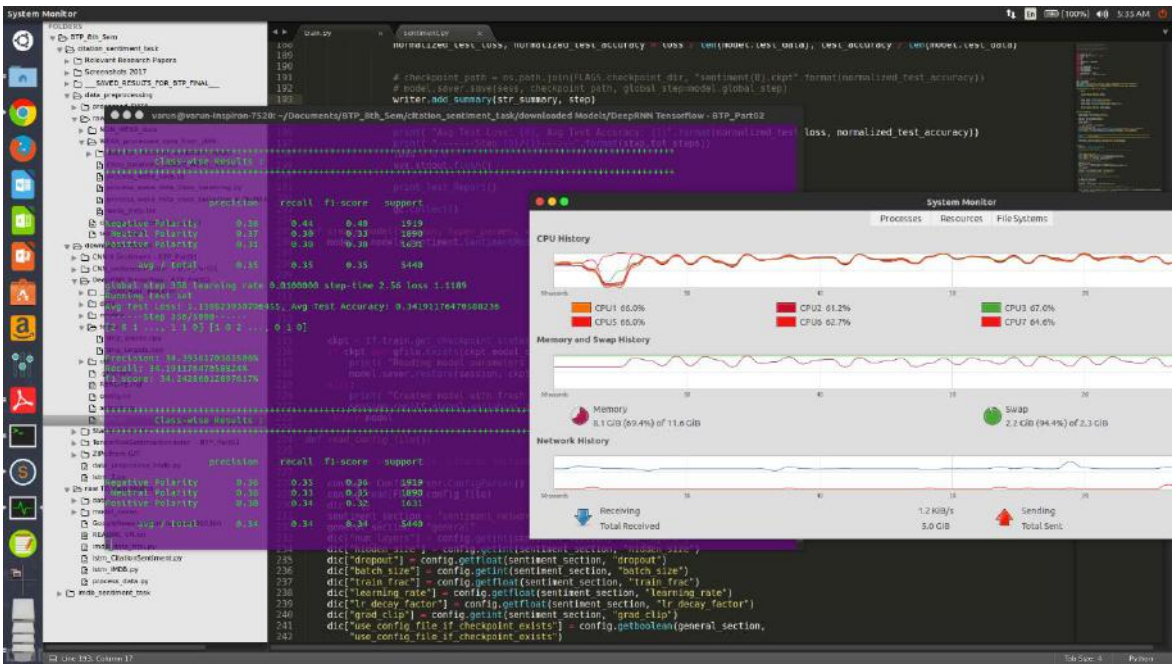


Fig. 6.1 Screenshot: *Neural Model being trained on CPU*

During the project tenure, although the experimentation was done upon several combinations of hyperparameter values and various different variants of the architectures tested, we present here only the significant ones and confidently state that the results obtained with other variants are nothing but random fluctuations of similar results. Some of the experimented results were very poor and bear no significance or value which we shall simply state below without mentioning the detailed results. It may be noted that the models have been developed based on the mentioned architectures, using the open source API libraries provided by Google, called, 'TENSORFLOW'.

Similar to the first phase, one major problem we encountered was the class-imbalanced dataset. For this we used ***"Class-Balancing Techniques"*** such as:

(1) *Over-sampling* of the under-sampled classes (*Positive* & *Negative*) and

(2) *Under-sampling* of the over-sampled class (*Neutral*)

Without such balancing of data, the results obtained were superfluous and misleading as in such a class-imbalanced scenario any trained model got biased to too much extent and learned to always output the majority proportion Class i.e. the *Neutral Sentiment Class* (class proportion: ~ 6000: 700: 200)

Also, it is quite important to note that the results obtained without class – balancing were close to 95% which can easily be achieved by any constant function due to high proportion of the predicted class in the test dataset.

Thus, in such cases, accuracy is NOT at all a justified metric to evaluate the model performance on any grounds. Thus, we report the performance of our model with the following experimental settings:

(A) *Trained on Class-balanced Dataset:*
The Dataset was partitioned into splits (train, test and validation) followed by Oversampling of the Positive & Negative Classes to bring them at 1: 1 ratio.

(B) *Evaluated using Metrics Independent of Class Label Statistics:* Such metrics include Precision, Recall, the F1-score, and the likes.

6.1 Classification Results – Deep Bidirectional RNN-based LSTMs

Although the term 'Deep' literally signifies the presence of an extended depth comprising several layers, however, it is quite an obvious fact that more the number of layers, more advanced the processor hardware is needed for training phases of execution. In fact, all the state-of-the-art achievements have been made on high-performance GPUs which is in general not easily accessible to the commons.

We didn't have access to Nvidia GPUs and ran our experiments only on the processors of our personal computers which severely limited and restricted us in terms of computing capabilities & processing power and bandwidth. In such a scenario, the best we could do is to set the number of Layers in the MultiRNN

Cell as 2. Thus, the "*Depth*" of our "*Deep*" LSTMs comprises of only two layers. The full set of hyperparameter values used are mentioned in table 6.1 below:

Table 6.1. Model Hyperparameter Settings (Bidirectional LSTMs)

Deep LSTM Hyperparameters	**Value**
# of layers in depth if LSTM cell	**2**
Size of each Hidden layer	**50**
Max Sequence Length	**200**
Max Vocabulary Size	**20000**

The following table 6.2 shows the classification results achieved during testing phase, after several long hours of CPU-bound training the model in units of multiple epochs:

$$Average\ Test\ Accuracy\ (Bi-directional\ LSTMs) : 61.213\ \%$$

Table 6.2. Citation Sentiment Classification Accuracy Statistics, Bi-directional LSTMs

Citation Sentiment Class Label	Precision	Recall	F1-Score	Support
Negative Polarity	0.87	0.53	0.66	1919
Neutral Polarity	0.59	0.80	0.68	1887
Positive Polarity	0.47	0.49	0.48	1634
AVERAGE MODEL ACCURACY	65.37 %	61.21 %	61.18 %	5440

6.2 Classification Results – Text – CNNs

Since we worked with limited resources and trained the model on a CPU (due to the lack of GPU), not many convolutional layers could be afforded. Thus, we worked with only one convolution layer. The full set of hyperparameter values used are mentioned in the table 6.3 below:

Table 6.3. Model Hyperparameter Settings (Text CNNs)

Text CNN Hyperparameters	Value
# of conv layers	**1**
# of filters in 1st layer	**128 x 3 (sizes)**
filter sizes	**3, 4, 5**
# of filters per filter size	**128**
Word Embedding Dimension	**128**

The LSTM architecture was tested on both the Pure Word Embeddings as well as Feature-concatenated Word Vectors. However, no apparent change was observed apart from a few random fluctuations pertaining to the random-split-dependent data noise effects. On the contrary, a significant difference in the results was observed in the case of Text-CNNs where the concatenation of feature-descriptors to the word embeddings added to the quality of the obtained classification results. The difference is apparent from the following two classification accuracy reports as shown in tables 6.4 and 6.5 respectively.

6.2.1 Classification with Pure Word Embeddings

The following are the classification reports when the model was trained and tested on the dataset composed of pure word vectors (only the word embeddings) without any concatenation of the descriptors of the features which were explicitly extracted from the sentence-level data format.

$$Average\ Test\ Accuracy\ (word\ vector\ based\ Text - CNNs): 53.367\ \%$$

Table 6.4. Citation Sentiment Classification Accuracy Statistics, Pure Word Vector based Text – CNNs

Citation Sentiment Class Label	Precision	Recall	F1-Score	Support
Negative Polarity	0.75	0.27	0.40	1575
Neutral Polarity	0.51	0.77	0.61	1904
Positive Polarity	0.50	0.51	0.51	1712

AVERAGE MODEL ACCURACY	57.8 %	53.17 %	51.12 %	5191

6.2.2 Classification with Feature-Equipped Word Vectors

The following are the classification reports when the model was trained and tested on the dataset composed of composite word vectors which were formed by concatenation of the descriptors of the features which were explicitly extracted from the sentence-level data format to the word embeddings. The details were mentioned in the Section 3.3.2, the figure 6.1 is presented again for reference:

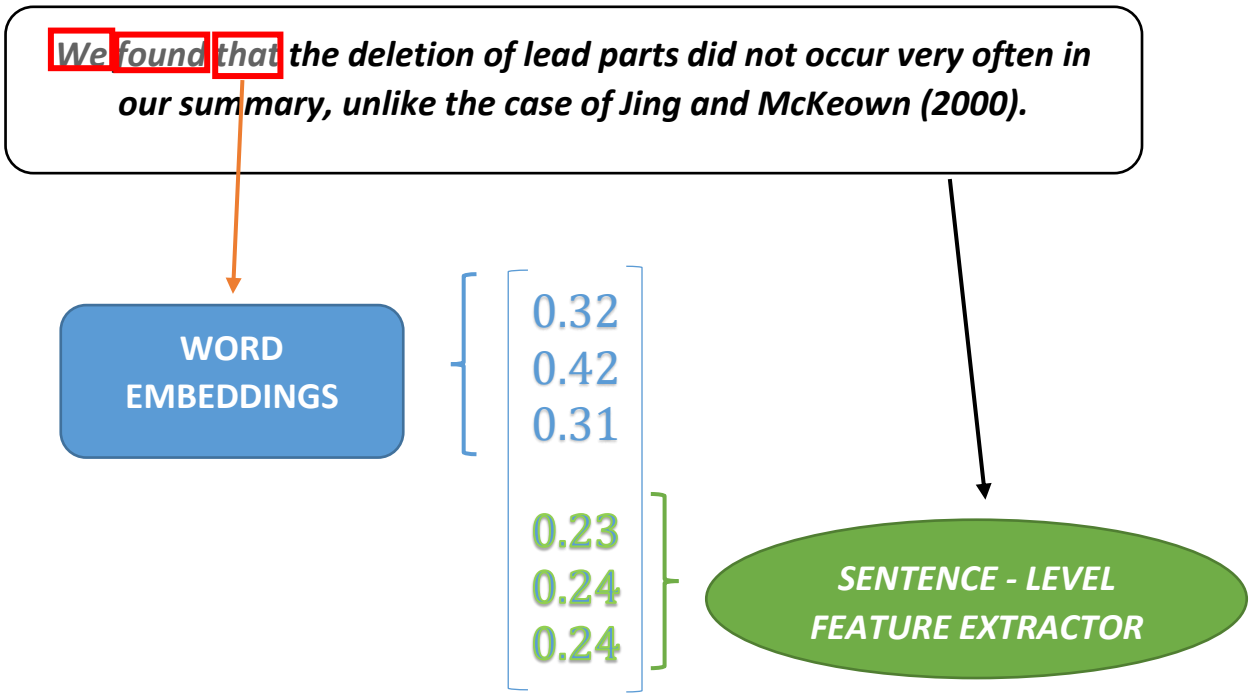


Fig. 6.1. Feature Extracted Word Vectors – Concatenation of Word Embeddings with Sentence-Level Feature Descriptors

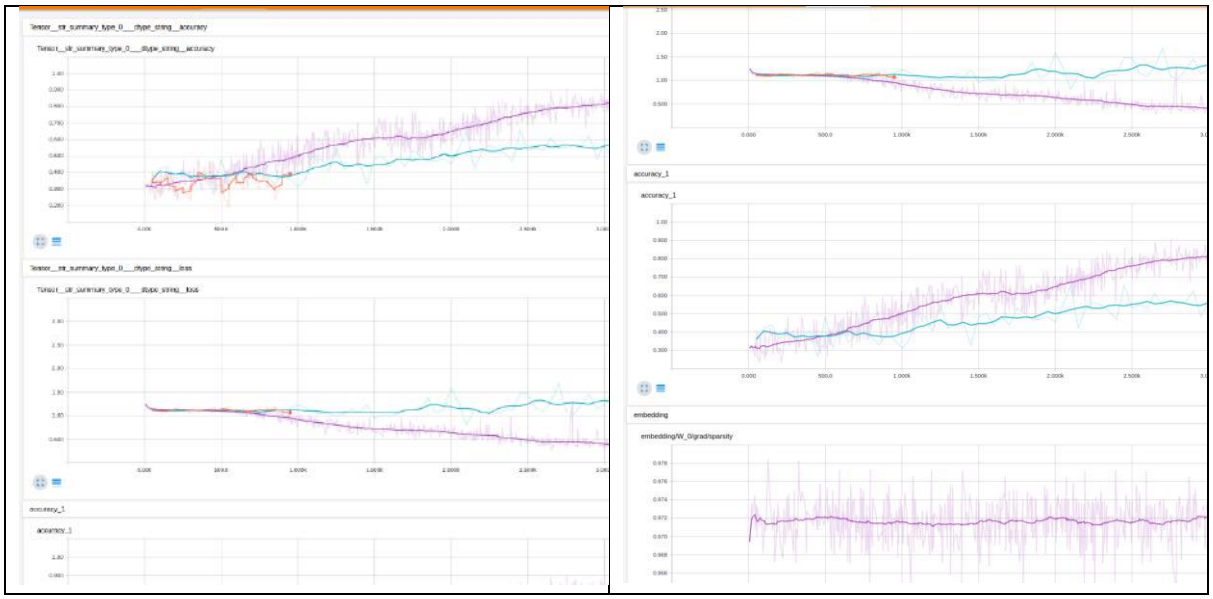
Equipping the word vectors with such details via concatenation of the explicitly extracted feature descriptors certainly demonstrated an improvement in classification quality as is demonstrated by the following evaluation report:

$$Average\ Test\ Accuracy\ (feature\ equipped\ word\ vector\ based\ Text - CNNs): 68.38\ \%$$

Table 6.5. Citation Sentiment Classification Accuracy Statistics, Feature- equipped Word Vector based Text - CNNss

Citation Sentiment Class Label	Precision	Recall	F1-Score	Support
Negative Polarity	0.81	0.82	0.82	1825
Neutral Polarity	0.64	0.72	0.68	1927
Positive Polarity	0.58	0.49	0.53	1688
AVERAGE MODEL ACCURACY	68.04 %	68.38 %	68.02 %	5440

In order to help debugging the code for model implementation, we had been collecting statistics about the various metrics and backpropagated gradients of the weights and biases at each layer and obtain their summary visualizations in the form of graphs, histograms and diagrams at regular intervals, via the help of Google facilitated TensorFlow library built-in utility tool, 'Tensorboard'. Such visualizations have been captured for both the neural learning architectures (LSTMs and Text CNNs) as assimilated in figures 6.2 and 6.3 respectively.



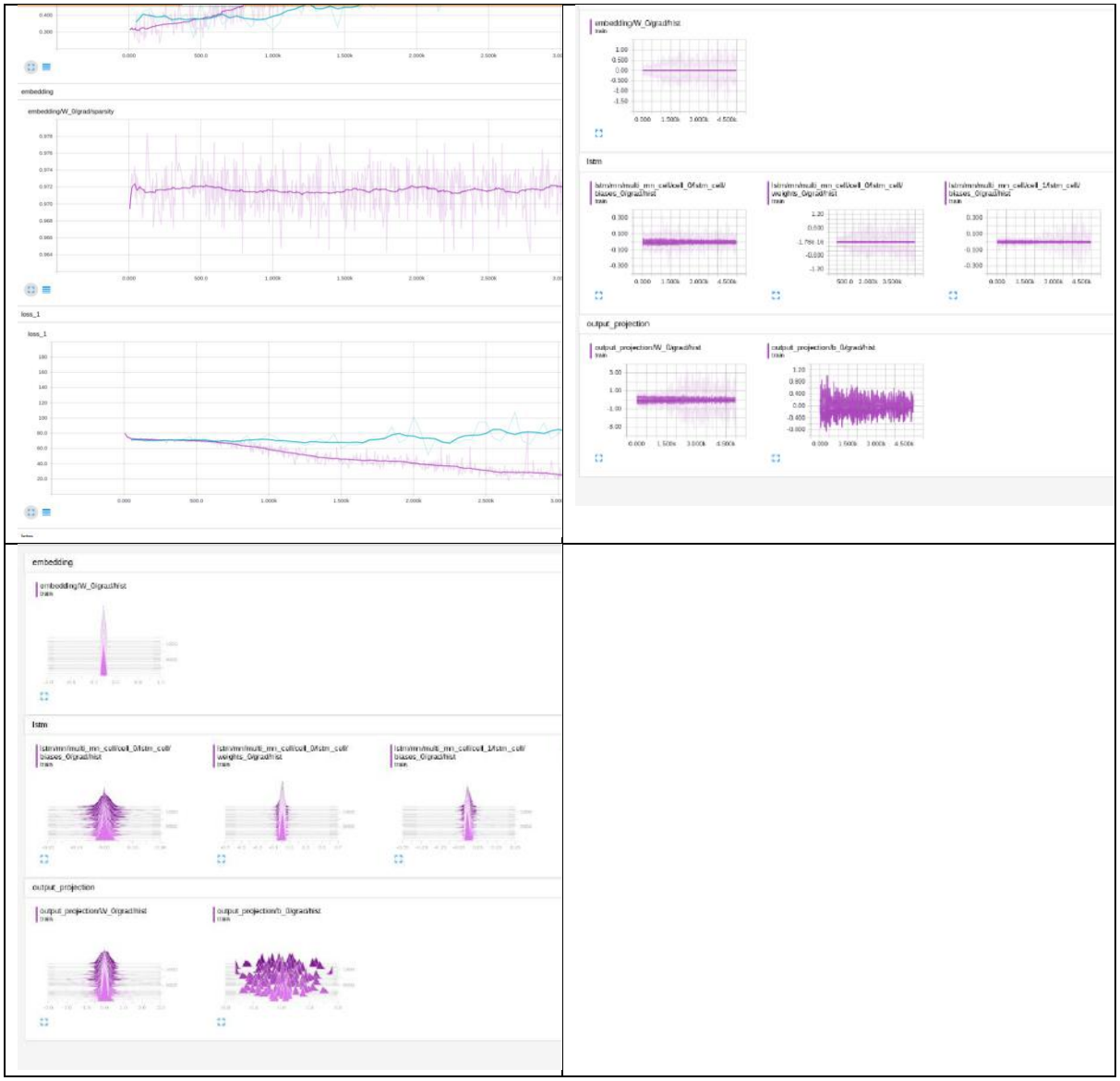


Fig. 6.2. Bidirectional LSTM Model gradient histograms, Accuracy and Loss visualizations in TENSOR-BOARD

6.3 Model Comparisons and Interpretation

After the commencement of mentioned project, there was no publicized literature upon the research domain of automatic citation classification, implemented with Deep Neural Learning Models, until a recent publication was made in November 2016 in the Proceedings of an International Workshop by Munkhdalai, et al. [11].

While we presented our results upon models of LSTMs as well as Text – CNNs, our work can be compared to their model performance comparisons including LSTMs and Attention Models, as illustrated in table 6.6 below.

Table 6.6. Deep Neural Net Learning Model Comparisons for different experimental settings

Classifier → Criteria ↓	**Bi-directional LSTMs (Our Model)**	**Text CNNs (Our Model)**	**Bi-directional LSTMs, Global Attention Models (Munkhdalai et al.)**
Feature Extraction	Word-level Linguistic features	Word-level Linguistic features	Pure word vectors
Classification Scheme	Citation Sentiment Classification	Citation Sentiment Classification	Citation Sentiment Classification
Annotation Corpus	Arthar, Awais, et al. (free, online)	Arthar, Awais, et al. (free, online)	Self-generated (not available online)
Test Input	Citation sentence	Citation sentence	Citation sentence + adjacent left and right sentences
Model Performance, Test (%)	68.02 %	61.18 %	76 % (approx.)

Chapter 7
Further Plan of Work

7.1 Future Scope in Our Work

With respect to the architecture frameworks, we plan to further implement the latest frameworks one of which is the ***Neural Attention Models*** recently published by [16] and [2] followed by several others.

These attention-based architectures have been successfully used and reported by [16] in CNNs for Vision – related task (Image Captioning) and also in RNNs by [2] for NLP– related task (Machine Translation).

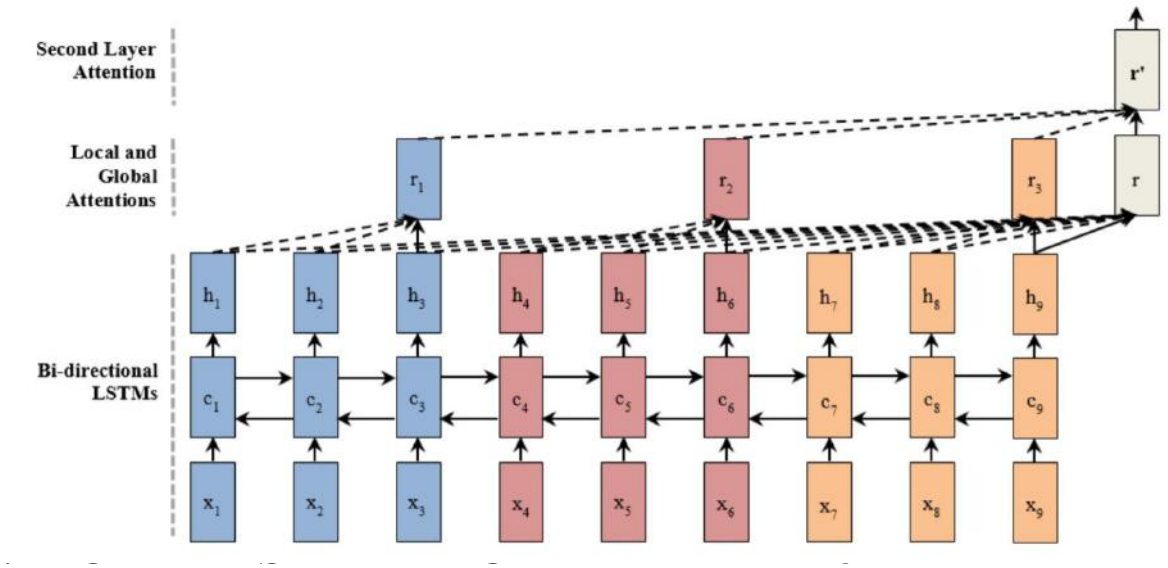


Fig. 7.1: Compositional attention network [11]

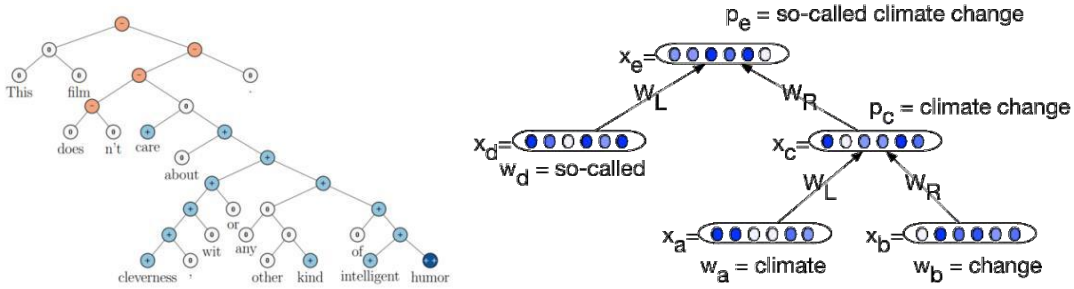
Attention Mechanisms in Neural Networks are (very) loosely based on the visual attention mechanism found in humans. Human visual attention is well-studied and while there exist different models, all of them essentially come down to being able to focus on a certain region of an image with “high resolution” while

perceiving the surrounding image in “low resolution”, and then adjusting the focal point over time.

With an attention mechanism, we no longer try encode the full source sentence into a fixed-length vector. Rather, we allow the decoder to “attend” to different parts of the source sentence at each step of the output generation. Importantly, we let the model **learn** what to attend to, based upon the input sentence and what it has produced so far. This is implemented by allowing to model the “attention” as a set of normalized probabilities that it assign to the input sentence / pixels in the dataset, which represent the ‘importance’ it chooses to give to the particular component while predicting the output at that time.

Another famous architecture that can be exploited for this task is the ***Recursive Neural Networks*** which happen to be a generalization of Recurrent Networks. This has been recently introduced by Socher et al. [14]

A recursive neural network (RNN) is a kind of deep neural network created by applying the same set of weights recursively over a structure, to produce a structured prediction over variable-size input structures, or a scalar prediction on it, by traversing a given structure in topological order.



This framework is mostly based on the idea that just similar to the way words can be represented in a d-dimensional embedding space, the phrases can also be distributed in a space where similar semantic phrases are placed close to each other. This is done by recursively composing the vector for a given phrase with those of its constituent word / phrases as building blocks of a recursive framework.

Stanford NLP group including researchers of [14] have already implemented a successful framework for sentiment classification of English sentences.

We even tested the results of our citation dataset on this framework which arbitrarily superfluous results as expected. This can certainly be attributed to the fact that citation sentiment analysis requires a lot more than just the meaning of phrases because of the inherently task-specific terminology used in the research works.

Thus, attempting to implement such a generic framework for Citation Classification remains a motivation and a challenging goal for us in the future, especially because of the burdensome annotation required for the target labels of such frameworks in which the target sentiment score has to be specified at every recursive level of the Parse Tree.

7.2 Future of ACA – The Road Ahead

The ***Automatic Citation Analysis (ACA)*** task has been and is being chased by several researchers of NLP. It can be extended in the directions of using different corpora, different annotation schemes, different feature sets and different classifiers. In the mentioned work, the classification task was implemented with the popular LSTM Architectures and can be extended using Global, Local and Compositional ***Attention Neural Frameworks***, as also reported by Munkhdalai, et al. [11] These word-vector based machine learning approaches also need to be implemented using input feature vector variants as discussed in the past works to arrive at performance comparisons among these models. Moreover, the hypothesis, they used that the size of the input layer to these neural learning models are only limited to citation sentences or at the most carried along the adjacent left and right sentences, needs further logical investigations.

The literature on the above deep neural net learning models also concluded at the point that Automated Citation Analysis still remains a challenge due to lack

of a large training annotated corpus. Although, they claimed to perform well upon their own developed annotation corpus, the testing of the same with past developed classification models would only adjudge their corpus as a benchmark or not. Thus, there is a need for a universally acceptable annotation corpora for performing benchmark experiments upon any type of machine learning task; so is the case in automation citation classification task. Very few annotation corpora are available online, among which the one released by Jochim and Schutze [13] follows MM (Moravcsik and Murugesan) annotation scheme of citation classification, while the one released by [1], which we have based our work upon, follows a different one. The need of the hour is to arrive at a universally acceptable corpus by the machine learning communities. This experimentation can again be tested for lots of feature vector variants like multi-word (n-gram) lexical features, linguistic features, structure features, location features, frequency features, sentiment features and many more.

As **Deep Neural Networks** require enormous amounts of training examples to be fed in, the 'Manual Annotation' process also need to be replaced with a speedy intelligent user-friendly process for quick generation of Data Corpus, whose baseline has been introduced during the preliminary stages of the work done. The scope of work in this direction would be to develop an Annotator Interface that shall be used on the 8736 available Citations labelled with sentiment to develop a complete Multi-Class labelled Data Corpus, expected to improve performance of deep learning neural net models thereafter.

Bibliography

[1] Awais Athar. Sentiment analysis of citations using sentence structure-based features. In *Proceedings of the ACL 2011 student session*, pages 81–87. Association for Computational Linguistics, 2011.

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

[3] Bilal Hayat Butt, Muhammad Rafi, Arsal Jamal, Raja Sami Ur Rehman, Syed Muhammad Zubair Alam, and Muhammad Bilal Alam. Classification of research citations (crc). *arXiv preprint arXiv:1506.08966*, 2015.

[4] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537, 2011.

[5] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

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

[7] Charles Jochim and Hinrich Schütze. Towards a generic and flexible citation classifier based on a faceted classification scheme. 2012.

[8] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[9] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[10] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[11] Tsendsuren Munkhdalai, John Lalor, and Hong Yu. Citation analysis with neural attention models. *EMNLP 2016*, page 69, 2016.

[12] Hidetsugu Nanba, Noriko Kando, and Manabu Okumura. Classification of research papers using citation links and citation types: Towards automatic review article generation. *Advances in Classification Research Online*, 11(1):117–134, 2011.

[13] Charles Jochim Hinrich Schütze. Towards a generic and flexible citation classifier based on a faceted classification scheme.

[14] Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, Christopher Potts, et al. Recursive deep models for semantic compositionality over a sentiment treebank. Citeseer.

[15] Simone Teufel, Advaith Siddharthan, and Dan Tidhar. Automatic classification of citation function. In *Proceedings of the 2006 conference on empirical methods in natural language processing*, pages 103–110. Association for Computational Linguistics, 2006.

[16] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057, 2015.