Understanding and Mitigation of Noise in Crowd-Sourced Relation Classification Dataset

A dissertation submitted in partial fulfillment of the requirements
for the award of the degree of

Doctor of Philosophy

Submitted by

Akshay Parekh

Under the supervision of

Dr. Ashish Anand & Dr. Amit Awekar

Department of Computer Science and Engineering
INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI
Guwahati 781039, Assam, India

April 2022

I would like to dedicate this thesis to Dada Gurudev and my loving family …

Acknowledgements

First and foremost, I would like to express my sincere gratitude to Dr. Ashish Anand, and Dr. Amit Awekar, my Ph.D. supervisors. They are excellent mentors and have always supported me in all my research endeavors giving me opportunities to work on research problems that excite me the most. Their assistance aided in the transformation of concepts into tangible goals. They consistently encouraged me to attend numerous seminars, tutorials, meetings, workshops, and conferences.

I am also thankful to my present and past doctoral committee members – Dr. Sanasam Ranbir Singh, Dr. Rashmi Dutta Baruah, Dr. Suresh Sundaram, and Dr. V. Vijaya Saradhi for providing valuable feedback during the research. I hereby express my gratitude towards Prof. Purandar Bhaduri and Dr. Chandan Karfa, for all their moral support. I also sincerely thank all the faculty members for their direct and indirect support.

I sincerely thank my mentor Balaji Ganesan from IBM Research Lab India, for hosting me for a research internship. Interactions with numerous brilliant minds – Hima Patel, Sameep Mehta, Berthold Reinwald, Saqlain Mustaq, Ayush Kumar, Abhay Shalghar, Ashwin Kanan, Raman Tehlan, Jaydeep during and after the internships always pushed me to strive for excellence in my research endeavors.

Also, I thank MHRD, the Government of India, for providing financial assistance throughout the Ph.D. program. I am thankful to funding agencies such as, Dept. of Biotechnology (project no. BT/COE/34/SP28408/2018) and Dept. of CSE for providing necessary computing resources used during the work.

I would also like to acknowledge the technical staff, system admins at IIT Guwahati and the ever-supportive administrative staff in the Dept. of CSE, Computer & Communication Centre, Academic Affairs, and Student Affairs, specially, Bhriguraj Borah and Nanu Alan Kachari. I am thankful to organization such as CoDS CoMAD 2020, IRIS 2020 and IIT Gandhinagar for providing travel support to attend conference and workshop.

I am particularly thankful to interns and collaborators, special mentions of Siddhant Srivastava, Hema, Aman Kumar, Mayank Baranwal, Shivang Dalal, Manan Gupta, Sayak Dutta, Pranav Gupta, Ajay Gahlot, Umang, and many more. I would like to thank all the amazing peers at IIT Guwahati – Arijit, Pawan, Pradeep, Harish, Ujjwal, Sanjukta, Divya, Aakansha, Dipojjal, Chinmay, Ghalib, Anasua, Palash, Aparajita, Swarup, Deepankar, Gourishankar, Gyanendro, Nidhi and many more to motivate, support, and celebrate with me throughout my Ph.D. journey. I would specially like to thank Abhishek for supervising and helping me throughout the PhD with various academic and non-academic endeavours. A vote of thanks to the seniors and juniors – Pranav, Saptarshi, Sunil, Akash Anil, Bala, Deepak, Brijesh, and many more.

I am fortunate to have several wonderful friends within and outside IIT Guwahati who were instrumental in providing me with strong support during my Ph.D. endeavor. Life never has a dull moment with them. I especially would like to thank Priya Ganguly, Dr. Suman Sharma, Priyank Rakhecha, Priyank Gandhi, Rajeev Pathak, Gaurav Jain, Ashutosh Mishra, Antariksh Dutt, Chirag Agarwal, Hussain Reshamwala, Mayank Vijayvargiya, Akshay Rohera, Himanshu Batra, Harsh Rungta, Nikita Hudia, Mahavir Parekh, Deepika Bishnoi, Prathamesh Raut, Ajinkya Mankar, Pranay Jain, Neelesh Shukla, Chirag Sodani, Akash Agarwal, Ankit Lath, Aditya Tiwari, Tanmay Panjwani, Anurag Prajapati, Krishna Kumar, Chandan Sharma, Sanga, Aditya, Shibasis, Anurag, and Athanu Paul. Thanks to the rest of my IIT Guwahati friends (including wall-mates and hostel-friends) who made my stay at IIT Guwahati lively and fun-filled.

I would like to thank my favorite set of people, the pillars of my strength — my parents, Dada Gurudev, my wife, my younger brothers and sister, and rest of my family members.

Finally, heartfelt thanks to IIT Guwahati for providing a beautiful serene campus with high-quality facilities. Last but not the least, a vote of thanks to the medical staff, security personnel, mess-staff, and cleaning-staff of IIT Guwahati.

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 50,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 50 figures.

Akshay Parekh
April 2022

Department of Computer Science and Engineering
Indian Institute of Technology Guwahati
Guwahati - 781039 Assam India

Certificate

This is to certify that this thesis entitled "**Understanding & Mitigation of Noises in Crowd-Sourced Relation Classification Dataset**" submitted by **Akshay Parekh**, in partial fulfilment of the requirements for the award of the degree of Doctor of Philosophy, to the Indian Institute of Technology Guwahati, Assam, India, is a record of the bonafide research work carried out by him under my guidance and supervision at the Department of Computer Science and Engineering, Indian Institute of Technology Guwahati, Assam, India. To the best of my knowledge, no part of the work reported in this thesis has been presented for the award of any degree at any other institution.

Date: April 2022
Place: Guwahati

Dr. Ashish Anand
Associate Professor
Dept. of C.S.E
IIT Guwahati

Dr. Amit Awekar
Associate Professor
Dept. of C.S.E
IIT Guwahati

Abstract

Text is one of the main sources of transmission of information between humans. It varies from personal chats to corporate e-mails, legal documents to scientific research articles, social media posts to blogs, and news bulletins to public announcements. The amount of such unstructured text in the open world is rising at an exponential rate. However, only a fragment of this information is available as knowledge for computer algorithms. Natural Language Processing (NLP) tasks such as Information Extraction (IE) and its sub-task Relation Extraction (RE) / Relation Classification (RC) can significantly improve the conversion from unstructured text to structured knowledge. Nonetheless, RE/RC is largely restricted due to the absence of high-quality datasets for training data-hungry deep neural models, which have shown excellent performance in other NLP tasks.

The overarching objective of this dissertation is to explore unconventional ways of improving large RC datasets and learning from them. Existing large RC datasets have few relations labels, ignore relations between relations, and are noisy and imbalanced. Creating a new dataset is not an optimal solution as it is a time- and cost-intensive process. The primary focus of this thesis is to analyze noise present in existing large-scale RC datasets and propose automated methods to mitigate some of the noise. In particular, work is focused on three main objectives, *(i)* characterizing the noise present in the dataset; *(ii)* exploring automatic and cost-sensitive approaches to reduce noise from the RC dataset; and *(iii)* analyzing the cost of reannotating them.

To this end, this dissertation makes three major contributions toward improving the RC from a large crowd-sourced dataset TACRED. The *first* work focuses on exploring the use of the relation between relation labels for reducing noise and improving RC models. Our preliminary analysis as well as some contemporary studies indicate that several incorrect relation labels can be identified by examining the corresponding `subject_entity` and `object_entity`. Based on this observation, we build a taxonomical relation hierarchy (TRH) from multiple KBs. We used it as a template for creating a similar TRH for TACRED

which is then used for exploring noise in positive instances and incorporating hierarchical distance between relation labels in RC models.

For our *second* contribution, we did a comprehensive evaluation of noise in the TACRED. All our analyses are based on SOTA RC models' predictions and performance. Following our findings, we investigate automated and cost-sensitive strategies for reducing noisy instances based on the nearest neighbors of examples with false-negative predictions and examples from a cleaner subset of TACRED. Empirical results have shown improved performance on the newly generated datasets.

In our *third* and final contribution, we utilize relation hierarchy for budget-sensitive reannotation of TACRED. We introduce the concept of a reannotation budget to provide flexibility on how much data to reannotate. We also proposed two strategies for selecting data points for reannotation. We performed extensive experiments using the popular RC dataset TACRED. We have shown that our reannotation strategies are novel and more efficient when compared with the existing approaches. Our experiments suggest that the reported performance of existing RC models on the noisy dataset is inflated. Furthermore, we show that significant model performance improvement can be achieved by reannotating only a part of the training data.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

Terms	Abbreviations
ACE	Automatic Content Extraction
AKBC	Automated Knowledge Base Completion
AL	Annotated Label
AMT	Amazon Mechanical Turk
BERT	Bidirectional Encoder Representations from Transformers
bi-LSTM	Bidirectional Long Short-Term Memory Network
CE	Cross Entropy
CNN	Convolutional Neural Network
DS	Distant Supervision
ES	Extrinsic Strategy
GCN	Graph Convolutional Network
HCE	Hierarchical distance scaled Cross Entropy
IE	Information Extraction
IR	Information Retrieval
IS	Intrinsic Strategy
KB	Knowledge Base
KBP	Knowledge Base Population
LOC	LOCATION
LSTM	Long Short-Term Memory Network
NER	Named Entity Recognition
NLP	Natural Language Processing
ORG	ORGANIZATION
PER	PERSON
PL	Predicted Label
QA	Question Answering
RC	Relation Classification

RE	Relation Extraction
RNN	Recurrent Neural Network
SOTA	State Of The Art
SVM	Support Vector Machines
TACRED	TAC Relation Extraction Dataset
TACRED-E	TAC Relation Extraction Dataset - Eliminated
TACRED-EN	TAC Relation Extraction Dataset - Eliminated Negative
TACRED-ENP	TAC Relation Extraction Dataset - Eliminated Negative Positive
TACRED-F	TAC Relation Extraction Dataset - Filtered
TACRED-FR	TAC Relation Extraction Dataset - Filtered & Relabeled
TACRED-R	TAC Relation Extraction Dataset - Reannotated
TACRED-RN	TAC Relation Extraction Dataset - Reannotated Negative
TACRED-RNP	TAC Relation Extraction Dataset - Reannotated Negative Positive
TRH	Taxonomical Relation Hierarchy

Chapter 1

Introduction

Currently, Deep Learning is the dominant paradigm in the solution for the Relation Classification (RC) task. With improvements in the Deep Neural Networks, the performance of Deep Learning based models for the RC task is also improving. Despite the progress in such models, the performance of RC models on the most widely used dataset is restricted to the range of 75% - 80% in F1-Score. Some of the recent papers attribute this restriction in model performance to the quality of the dataset. Furthermore, they observed that the models' performance can be improved by reannotating either part or the entire dataset. Their work, on the other hand, is heavily reliant on human efforts, either through experts or crowd-sourcing. To this end, in this dissertation, we explore automated and cost-sensitive approaches for analysing and reducing noise from the large RC dataset TACRED. In our first work, we create a label hierarchy and use it to improve dataset quality and to modify cross-entropy loss function for model optimization. Then, in the second work, we thoroughly analyse state-of-the-art models' predictions and performance to characterize noise in the dataset and further evaluate two different strategies for reducing the noise. Finally, in the last chapter, we introduce the concept of a reannotation budget, to efficiently reannotate datasets with minimal effort.

1.1 Motivation

Since the introduction of the internet, the amount of digital data in the form of news articles, blogs, emails, government documents, chat logs, research articles, etc., is increasing at an exponential rate. Given the heterogeneous and unstructured nature of data, it will be of some

value only if it can be tagged and annotated efficiently. Tagging such large amounts of data is not possible manually. Hence, there is a need for a system that automates the extraction and annotation of structured information from text. This structured data can later be used in various real-world applications, such as:

- Search engines retrieving set of relevant documents,
- Virtual assistants answering facts based queries using online encyclopedia such as Wikipedia,
- Business domain-specific applications like, analysis, reasoning and decision making from news and reports for businesses,
- Biomedical domain-specific applications such as, analysing drug interaction, protein reactions based on bio-medical research articles and clinical trials.

Information Extraction (IE) is the task of extracting precise information from large unstructured text corpora and making it available in annotated structured form. IE systems take in large volumes of documents as input and output a structured resource of knowledge or facts. Such knowledge resources are often called "knowledge bases" (KBs). Some of the most popular KBs are DBpedia [1], Wikidata [2], and YAGO [3].

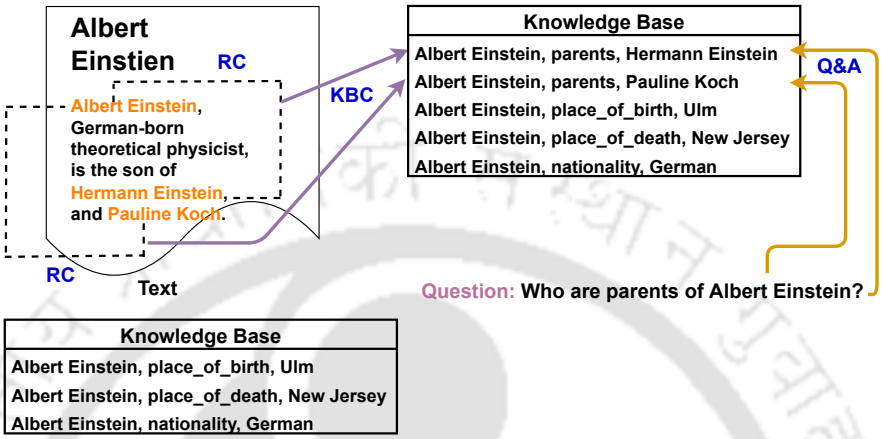
IE uses various linguistic, natural language processing (NLP) and text mining tasks and techniques. It includes text normalization, finding and classifying real-word entities, connecting those entities, and finding and classifying relations between the identified entities. Among all such tasks, **Named Entity Recognition (NER)** [4] and **Relation Extraction (RE)** [5] are the two important tasks.

Named Entity Recognition is the task of identifying, and classifying real-world entities mentioned in the given text. For example, in Figure 1.1 *Albert Einstien*, *Hermann Einstien*, and *Pauline Koch* are entities of type *Person* in the sentence.

Relation Extraction is the task of extracting the relation mention between a pair (or more) of entities present in the text. The identified relation is classified into one of the relation labels. This process is known as Relation Classification (RC). For example, in Figure 1.1, entities *Albert Einstein*, *Hermann Einstein*, and *Pauline Koch* are extracted from the given sentence. Relation between them is identified and classified as relation `parent`. Even though RE and RC differ from each other, they have been synonymous in the literature. The focus of this thesis is on the Relation Classification (RC) problem.

Relations between entities are a crucial component of structured knowledge. It can be mentioned in numerous different ways in the text. Thus, the main objective of RC is to

Fig. 1.1 Example of RC and it's potential applications. The relation triples (*Albert Einstien*, *per* : *parent*, *Hermann Einstien*) and (*Albert Einstien*, *per* : *parent*, *Pauline Koch*) from the sentence in the Figure, is first identified using RC and then it is populated in the Knowledge Base. Populated Facts are further used by Question Answering (Q&A).



devise methods that can efficiently identify relations of interest, no matter the way they are mentioned. The identified relation label along with the entity pair is generally arranged as a *triple*. This triple is finally used in populating facts in KBs (Automated Knowledge Base Completion (AKBC)). Such triples can then be used by NLP tasks such as question answering (illustrated in the Figure 1.1).

Relation Classification is generally formulated as a supervised learning problem. Earlier approaches relied heavily on models such as SVMs and kernel methods, whereas recent works heavily rely on modern deep neural models such as LSTMS and transformer-based models. Training these models requires a large amount of quality annotated datasets. Generating quality annotated datasets at such a large-scale has always been challenging due to the time and cost involved. Even after manual annotation, large datasets are often noisy.

For downstream applications, noisy nature of dataset is a matter of concern. Specially, for application domain such as finance and medicine the consequences can be immediate and irreversible. Extracting incorrect information from a document in the financial domain can cause exceptional business losses. Whereas, in the case of clinical document analysis, it can lead to erroneous decisions that could directly impact human life.

The majority of existing research focuses on either generating new datasets in cost- and time-effective ways or reannotation involving experts. In this thesis, we focus on identifying the reason behind the noisy nature of the dataset. How can we reduce such noise without

relying much on experts or manual efforts? What percentage of dataset reannotation is enough for efficient model training?

1.2 Challenges

Applications that rely on relation classification highly depends on the quality and precision of the extracted relation instances. Therefore, the high performance of relation classification becomes crucial for their success. In practical scenarios, RC suffers from following set of challenges:

1. **Shallow set of relation labels:** Existing RC benchmark datasets like ACE and TACRED rely on task-specific corpora such as multilingual automatic content extraction and TAC KBP slot-filling, respectively. Consequently, the set of relation labels gets restricted to the underlying corpus. Moreover, deriving relations between the relation labels becomes impossible because of the handful of relation labels.

2. **Ambiguity in relations:** Relation mention in a sentence is often learned based on the context between the entity mentions. But determining the boundary between certain relation labels gets complex due to the overlapping context. For example, relations *country of birth* and *countries of residence* often get confused, as in most cases, a person's country of birth is amongst one of their countries of residence.

3. **Long-tail distribution:** The performance of the model depends on the quality and quantity of training samples for each class label. However, finding a proportionate number of instances for each class has always been a challenge while creating large datasets.

4. **Noisy Dataset:** RC is a classification problem and therefore requires a large dataset for efficient training of deep neural models. Generating a large dataset for supervised learning requires experienced annotators and subject-experts. Finding such experts to annotate hundreds of thousands of sentences could be a very costly process. Hence, researchers rely on annotation tools such as Amazon Mechanical Turk (AMT) to employ multiple annotators. Inexperience and lack of expertise among those annotators lead to noise in the dataset. For example, Alt et al. [6] in their work, have highlighted the noisy nature of TACRED [7], largest and most-popular RC dataset.

5. **Inflexible Reannotation Strategies:** All efforts to reannotate the noisy RC dataset involve either linguistic experts reannotating a fixed set of the dataset or inexperienced crowd-sourced annotators reannotating according to modified guidelines.The problem with the former approach is that all the noisy instances cannot be annotated due to the cost involved in hiring linguistic experts. In the latter case, reannotating the entire dataset will take the same amount of time and money as creating a new dataset and may not even be required.

1.3 Problem Description

Large crowd-sourced datasets often exhibit one or more of the challenges mentioned in the previous section. Some recent studies have suggested that label noise is the prominent reason why RC model performance is stuck at a certain threshold on TACRED. Due to the incorrect annotations, the accuracy of the models has decreased, making the learning process more complex.

Some of the earlier work has been devoted to studying the noise and reannotating the dataset. However, those efforts heavily rely on expert analysis and crowd-sourced efforts, which ignore the inherent relation definition and model predictions, and, finally, do not allow for a flexible number of re-annotations.

In this thesis, our primary objective is to explore automated, cost-efficient strategies and heuristics to:

- Identify the root cause of noise in the dataset.
- Reduce noise from the dataset by either elimination or reannotation.
- Improve model optimization, making it robust for noisy dataset.
- Identify minimum number of reannotation enough for efficient model training.

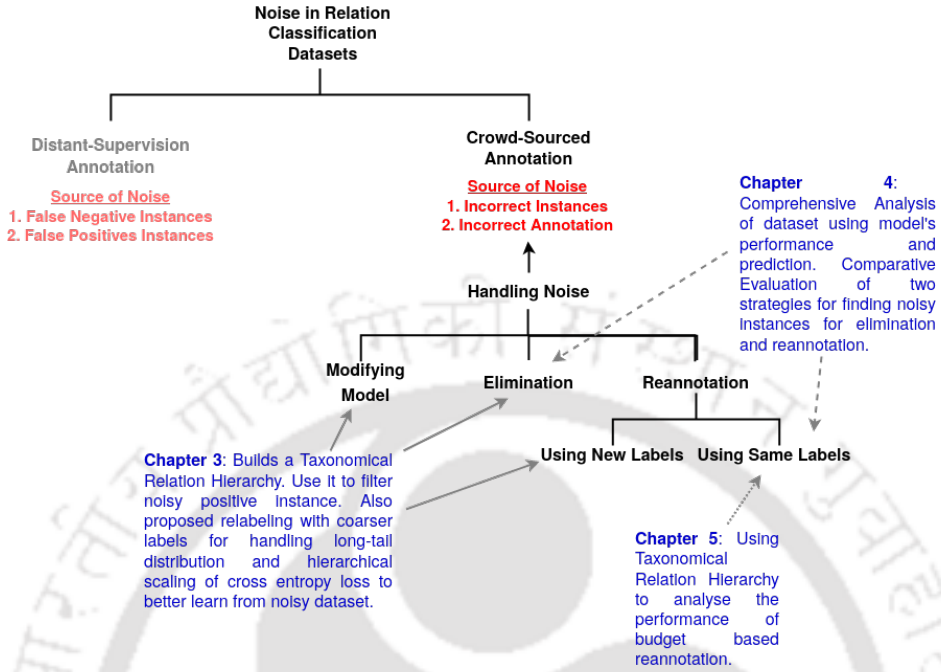
1.4 Contribution

In this dissertation, we make three contributions towards the identified objectives. We used the most-widely used and the largest crowd-sourced RC dataset TACRED [7] for this purpose. The graphical overview of our work is presented in Figure 1.2.

1.4.1 Contribution 1: Taxonomical Relation Hierarchy and it's application in Relation Classification

Our preliminary analysis, as well as some contemporary studies [6, 8], indicate that several incorrect relation labels can easily be identified by examining the corresponding `subject_entity` and `object_entity`. Based on this observation, the first contribution introduces a framework for building taxonomical relation hierarchy (TRH) from multiple KBs. We used this framework as a template to build a similar TRH for TACRED, which is

Fig. 1.2 An Overview of thesis contribution.



then used for exploring noise in positive instances and incorporating hierarchical distance between relation labels in RC models. In this work, we explore how we can use relation hierarchy to address all of the identified challenges in the earlier Section 1.2.

In our first contribution, we explored multiple KBs like Wikidata, DBpedia, and Wikipedia Infoboxes to compile a list of all possible relations between entity types *Person*, *Organization*, and *Location*. We collected a set of 623 canonical relations and arranged them in a taxonomical relation hierarchy following the proposed framework. Further, we used that relation hierarchy as a template for creating a similar relation hierarchy for 41 TACRED relation labels. We used the generated TRH to eliminate 164 ambiguous instances from the dataset. To address the long-tail distribution of relation labels and ambiguity in relation boundaries, we relabeled 17 finer relations with their coarser relation labels, improving model performance by an average of 1.3 percent across four baseline models. Finally, we used hierarchical distance to scale cross-entropy loss (CE Loss) and proposed *hierarchical distance scaled cross-entropy loss (HCE Loss)*. Our proposed approach HCE Loss improved SpanBERT's [9] performance by 3% in F1-Score.

In this work, we have just used TRH for filtering and relabeling some of the instances to reduce noise, long-tail distribution, and relation ambiguity problems. However, TRH can further be used for augmenting instances of relation labels using templates learned from the sentences and valid triples of either coarse or fine relations from knowledge-bases.

1.4.2 Contribution 2: Model-based Characterization and Reduction of noisy instances from RC dataset

From our first contribution, we realize that of all the challenges mention in the Section 1.2, noisy nature of data posed the most threat. Thus towards the second contribution, we did a comprehensive evaluation of noise in the TACRED and explored automated and cost-sensitive strategies for reducing noisy instances.

Following TRH in our previous contribution, [10], we could only eliminate a few noisy positive instances. Thus, the true picture of noise in the TACRED dataset was still unclear. Therefore, we explore the model-based characterization of noise present in the TACRED dataset and two strategies to handle potential noisy instances. Analyses of model prediction results indicate that the incorrect labelling of instances as *NO_RELATION* class or negative relation is predominantly responsible for the noise in the data. Hence, this work proposes two different strategies for identifying potential noisy negative relation instances for elimination and reannotation. The first strategy, **intrinsic strategy (IS)**, is based on finding the nearest neighbour to the model's false negative prediction. Whereas, the second strategy, **extrinsic strategy (ES)**, requires a subset of clean TACRED instances. Models trained on a dataset with elimination based on the intrinsic strategy show improvement when models are evaluated on the cleaner version of the test set. The performance of the models significantly improved with the extrinsic strategy for both the eliminated and reannotated datasets. Furthermore, identifying noisy instances among positive relation classes using the extrinsic strategy shows further improvement in the models' performance.

1.4.3 Contribution 3: Budget-Sensitive Reannotation of noisy RC dataset

In our third and final contribution, we address the last challenge, *"inflexible reannotation strategies"*, mentioned in section 1.2. We utilize relation hierarchy for budget-sensitive reannotation of TACRED.

Given the large size of datasets, reannotation of the complete dataset is not always possible due to the time and cost involved. In this work, we have introduced the concept of a reannotation budget to provide flexibility about how much data to reannotate. We also proposed two strategies for selecting data points for reannotation. Our approach capitalises on the taxonomic hierarchy of relation labels from our earlier work, [10]. For each data point, we compute the graph distance between the actual label provided in the dataset and the predicted label using an ensemble of RC models. Data points with a higher value for

this distance are given higher priority for the reannotation task. We performed extensive experiments using the popular RC dataset TACRED. We have shown that our reannotation strategies are novel and more efficient when compared with the existing approaches. Our experiments suggest that the reported performance of existing RC models on the noisy dataset is inflated. The F1 score of these models drops from the range of 60%-70% to as low as below 50% when tested on clean test data generated using our reannotation strategy. Furthermore, we show that model performance improvement can be achieved by reannotating only a part of the training data.

1.5 Thesis Outline

The thesis (Figure 1.2) is organized as follows.

Chapter 1 presents the background, motivation and problem formulation.

In Chapter 2, we first describe essential deep learning models, basic NLP terminologies, and concepts associated with the Relation Extraction and Relation Classification tasks. Then we provide a comparison between different RC approaches and available datasets.

In the next three chapters, we present the three contributions of the thesis. Relevant literature reviews are presented in the corresponding contribution chapters. In Chapter 3, we present the creation of a taxonomical relation hierarchy and its adaptation to TACRED for improving RC. In Chapter 4, we present a comprehensive analysis of noise in the RC dataset TACRED and also explore two approaches for finding noisy instances for elimination and reannotation. In Chapter 5, we present two strategies using hierarchical distance for budget-sensitive reannotation of RC datasets.

Finally, in Chapter 6, we conclude and discuss future research work.

Chapter 2

Background

This chapter serves as a foundation for the following chapters. We first introduces basic modelling approaches used in NLP. We then provide an overview of common natural language processing (NLP) tasks.

Subsequently, we describe essential terminologies and concepts, followed by describing the main task of this thesis: relation classification. We describe different problem formulation, modeling approaches, and datasets specific to RC task.

2.1 Modelling Approaches

In this section, we'll go through the deep neural models that are commonly used in NLP tasks, specifically RC, and that will be referenced frequently throughout this thesis.

Word Embeddings

When using neural models for NLP tasks, each word w_i in a sentence s needs to be mapped to a vector of real numbers. This real-valued vector x_i for a word w_i is known as *word embedding*. The word embeddings for entire *vocabulary* V for a dataset D is stored as an *embedding matrix* $E \in R^{|V| \times d}$. A sentence s, consists of a sequence of words $[w_1, w_2, ..., w_n]$ is represented as a sequence of word embeddings $[x_1, x_2, ..., x_n]$ before giving them as inputs to a neural network. Some of the most commonly used static word embeddings are Word2Vec

[11], Glove [12], and FastText [13]. In this thesis, we have used Glove embeddings unless explicitly mentioned otherwise.

Recurrent Neural Network (RNN)

RNNs [14] are type of neural networks in which the output from the previous step is used as input in the current step. In traditional neural networks, inputs and outputs in the adjacent steps are independent of one another. However, in some circumstances, such as, when predicting the next word of a phrase, the information from prior words are necessary, and so the previous words must be remembered. RNNs have the concept of 'memory' that helps them store the states or information of previous inputs to generate the next output of the sequence.

However, due to the problem of vanishing or exploding gradients, the RNNs faces difficulty learning over longer sequences. At each stage of the forward pass, the hidden state is multiplied by the weight matrix. During the backward pass, or back-propagation, the gradients are multiplied with the same values at each step, which can cause the gradients to explode or vanish. The model is unable to learn in both circumstances. One of the reasons for developing long-short term memory networks is to address this issue.

Long Short Term Memory Network

LSTMs [15] are popular RNN architecture introduced to address the problem of long-term dependencies. That is, if the previous state that is influencing the current prediction is not in the recent past, the RNN model may not be able to accurately predict the current state. To remedy this, LSTMs have "cells" in the hidden layers of the neural network, which have three gates– an input gate, an output gate, and a forget gate. These gates control the flow of information which is needed to predict the output in the network. The LSTMs were designed to overcome the vanishing gradients problem that occur when training traditional RNNs.

Bidirectional Long Short Term Memory network (biLSTM): To comprehend the context efficiently in some NLP tasks, it is necessary to understand not only the previous words in the sequence, but also the forthcoming words. A biLSTM is a sequence processing model that achieves this by using two LSTMs: one that takes the input forward and the other that takes it backwards. This effectively boosts the amount of data available to the network.

We have used LSTM, biLSTM and position-aware LSTM [7] throughout our works.

Convolutional Neural Network

CNNs [16] are another type of neural networks most commonly used for computer vision related tasks. In this dissertation we have used CNN adapted for NLP tasks [17]. Typically, a convolutional layer slides filters of different window sizes over the concatenated input word embeddings. Each filter weight computes a new feature for each window of k words. Finally *min*, *mean*, or *max* pooling is applied to get the most essential feature by reducing the feature map.

We have used CNN proposed by Nguyen et al. [18] for relation classification in all our works.

Transformers

The transformer [19] is a more modern neural network architecture that was inspired by a desire to replace RNNs' fundamentally sequential computation with a more parallelizable method based on (self-)attention [20]. The transformer layer, consists of two sub-layers: multi-head self-attention and a position-wise feed-forward neural network, as the basic building block. Around each of the two sub-layers, a residual connection [21] is used, followed by layer normalisation [22]. This architecture is used by many state-of-the-art methods for language modelling and transfer learning, such as the OpenAI GPT [23] and BERT [24].

In our works we have used transformer-based models such as BERT [24], RoBERTA [25], and SpanBERT [9]

2.2 Natural Language Processing

Natural Language Processing (NLP) allows computers to understand text in the same manner as humans do. This is usually expressed as a set of annotation tasks, in which a text is mapped to linguistic structures that indicate its meaning. We used machine learning based tools to learn such mappings.

Now I'll provide a quick review of the tasks that will be mentioned throughout this thesis. For each task, Figure 2.1 presents an example sentence with annotations. Sequence labelling

Fig. 2.1 Annotations for natural language processing tasks.

TASK	ANNOTATION						
Sentence	New Delhi is the capital of India.						
Tokenization	New	Delhi	is	the	capital	of	India
Part-Of-Speech Tagging	NNP	BBP	VBZ	DT	NN	IN	NNP
Named Entity Recognition	B-LOC	I-LOC	O	O	O	O	B-LOC
Dependency Parsing	compound	nsubj	root	det	attr	prep	pobj

tasks such as part-of-speech tagging, named entity identification, and dependency parsing assign an output y_i to each word w_i.

Tokenization: Tokenization is breaking a sentence into a list of tokens. Tokens can be word, character, or sub-words.

Part-of-speech (POS) Tagging: POS tagging assigns each word in a text its corresponding part-of-speech tag. A part-of-speech is a group of words with common grammatical features, such as nouns, verbs, adjectives, adverbs, pronouns, prepositions, conjunctions, and so on. A word can also be more than one part-of-speech when used in different context. For example, in sentence *Lucy was making lunch for Francis*, "make" is in the form of *verb*, whereas in the sentence *What are the make, model, and year of this car?*, it is in the form of *noun*.

Named Entity Recognition (NER): NER tagging is one of the sub-task of IE. It involves identifying real-world or abstract object mention, in a given sentence and classifying it with one of the categories. The identified real-world objects are known as *named-entities*. Some of the most common named-entities types are PERSON, LOCATION, ORGANIZATION, and so on. Generally, the types depend upon the underlying application. For example, biomedical texts will focus on entity types such as GENES, DRUGS, PROTEINS, DISEASES, etc. NER is an important step for relation extraction.

Dependency Parsing: Dependency parsing determines the dependency structure, or dependency parse, of a sentence. The parse is made up of words connected by directed links that represent the sentence's grammatical structure. Each link connects a head word to its dependent (the child), which alters the head according to the connection's syntactic relationship. Many applications use dependency parsing, including co-reference resolution, question answering, and relation extraction.

2.3 Knowledge Bases

A Knowledge Base (KB) is a collection of data that represents real-world facts. Unlike traditional databases, which are represented as tables. Knowledge Graphs (KG) are a popular approach to store KBs on a computer in a graph structure format. The graph's nodes represent entities, while the labelled edges reflect relationships between them. A KG can be represented as a set of (subject, predicate, object) (SPO) triples, where subject and object are entities, and predicate is a binary relation type. Presently, various KBs available, the most notable ones that we have used in our work are:

Wikipedia Infobox

Wikipedia is a multilingual, web-based, free-content encyclopedia project supported by the Wikimedia Foundation. It is based on a model of openly editable content. Wikipedia's articles provide links designed to guide the user to related pages with additional information. Since its creation in 2001, Wikipedia has grown rapidly into one of the largest reference websites. As of 8 May 2022, there are 6,495,516 articles in the English Wikipedia [1].

Wikipedia infobox is a fixed format table added to the Wikipedia page. It summarizes important facts and statistics of a particular page in a tabular format. The information it contains are comparable, concise, materially relevant to the subject and already cited somewhere in the article.

[1] https://en.wikipedia.org/wiki/Wikipedia:Size_of_Wikipedia

Freebase

Freebase is large collaborative Knowledge Base, initially developed by MetaWeb in 2007. It was later acquired by Google in 2010 and was shutdown in 2016. Metaweb described Freebase as "an open shared database of the world's knowledge". Freebase contained data from various sources such as Wikipedia, Notable Name Database (NNDB), Fashion Model Directory, MusicBrainz and data contributed by it's users. It's 2015 snapshot is available for download and contains a total of 1.9 billion triples.

DBpedia

The DBpedia project was started in 2007 jointly by the Free University of Berlin and the University of Leipzig to automatically extract structured information contained in Wikipedia, such as infoboxes, category information, geo-coordinates, and external links. It contains around 600 million triples in the English language and around 2.5 billion triples in other languages combined.

Wikidata

Wikidata [2] is a collaborative edited Knowledge Base operated by WikiMedia Foundation. It aims to provide a common source of data which can be easily accessed by WikiMedia projects and by anyone under public domain license. Wikidata is a document-oriented database, focused on items. Each item represents a topic and is identified by a unique number. An item can have one or more statements. Information is added to items by creating statements, in the form of key-value pairs, with each statement consisting of a property (the key) and a value linked to the property. Wikidata currently contains 97,626,220 items and more than 4000 properties[2].

2.4 Relation Extraction & Classification

Relation Extraction is the task of detecting the *semantic relation mention* between the *entity mentions* in the text and was first formulated at Message Understanding Conference (MUC)

[2] https://www.wikidata.org/wiki/Wikidata:Statistics

1998. Relation r can be defined on any number of entity. It is defined as a tuple with entities as its element.

$$r = (e_1, e_2, ..., e_j)$$

where, e_i is an entity and $r \in R$, the set of all possible relations. Relation can be n-ary tuple, but in this dissertation, we focus only on binary relation. For example, **Founder_Of(Steve Jobs, Apple)**, where **Founder_Of** is the relation between entities *Steve Jobs* and *Apple*.

Relation Classification is task of classifying the identified relation mention with one of the relation class. Relation classification is usually defined as supervised learning task that learns a function f which maps input features x derived from a sentence s to a relation label $r \in R$. This can formally be defined as, given a sentence s annotated with a pair of entities (e_1, e_2), an RC model tries to predict the relation r between subject entity e_1 and object entity e_2 (i.e. $f(x, e_1, e_2) \mapsto r$), where r is either one of the relation from a fixed set of relations (R) or `no_relation`.

In this work, we consider the function f as a deep neural network (DNN) with softmax as output layer.

$$p(\hat{y}/s, (e_1, e_2)) = Softmax(Wh^* + b)$$

$$\hat{y} = argmax_r(p(\hat{y}/s, (e_1, e_2)))$$

where h^* is output from last hidden layer of DNN, W & b are corresponding *weights* and *biases* and $\hat{y}$ is the output relation. The training objective is to score the correct $r \in R \cup \{\phi\}$ over all incorrect relation, using loss function such as cross entropy loss given by,

$$J = -\sum_{i=1}^{n_r} t_i log(y_i) \tag{3}$$

where $t \in \mathbb{R}^{n_r}$ is one-hot encoding to represent the ground-truth and $y \in \mathbb{R}^{n_r}$ is the estimated probability for each relation label by softmax.

Some of the popularly used datasets and methods are discussed as follows:

2.4.1 Datasets

Majority of the earlier RC models have used SemEval-2010 task 8 [26] and ACE multilingual datasets [27]. However, these datasets are not relevant for large-scale relation classification setting as they are small in size and address only hand-full of relations.

Distant Supervision [28, 29] has provided a way to address the absence of a large-scale RE dataset at low cost by leveraging a KB. But, in practice, the generated dataset is highly noisy. Noise is primarily due to either incomplete nature of knowledge bases (false negative instances) or due to incorrect mapping with entity mentions (false positive instances).

TACRED [7], a large crowd-sourced RC dataset, has become popular in the recent years. It contains more than 100 thousand instances manually annotated by crowd-sourced workers. For all empirical analysis, we have used this dataset.

However, some studies [6, 8] have suggested that it contains significant annotation errors. Alt et al. [6] trained a multitude of models to score the instances based on misclassification. They end up reannotating around 2500 instances from test and dev set with the help of linguistic experts (referred as TACRev). Whereas the other work from Stoica et al. [8] modified the annotation guidelines and reannotated the entire dataset with the help of crowd-sourced workers (referred as ReTACRED).

2.4.2 Methods

The literature shows following established approaches of the relation classification methods:

2.4.2.1 Rule-based Methods

Early rule-based methods [30] use hand-crafted extraction rules or templates for different relation labels, similar to regular expression. An input is processed in subsequent steps via pattern matching on logical forms extracted from the input. Some of the later methods [31, 32] rely on sentence templates, derived from supervised data. The templates include surface-level word representations, dependency parse grammar, and entity types. Once the templates are obtained, they can be used to extract relations from text via explicit pattern matching.

An advantage of pattern-based methods is that it allows humans to handcraft rules and easily inspect the systems state, which is often required in industrial applications. However,

these approaches are not scalable, hence majority of the research got shifted towards machine learning and deep learning methods.

2.4.2.2 Statistical & Machine Learning Methods

Statistical machine learning methods involve formulating complex mathematical models on available dataset. These methods required extensive feature engineering before-hand, where the lexical, semantic, syntactic, and entity-specific features are extracted from the input text. The commonly used features are: dependency parse tree, entity mentions, entity distance, POS tags, and so on.

Some of the earlier used approaches were based on SVM [33–35], kernel methods [36–39], and maximum entropy models [40]. These methods exhibit scalability and superior performance. However extensive feature engineering is their major limitation.

2.4.2.3 Deep Neural Methods

Recent relation classification methods use neural networks to model the problem, which has resulted in a significant improvement in the performance, owing to their capacity to learn meaningful input representations automatically and successfully model sequential data. Socher et al.[41] were the first to apply neural networks to the task using a matrix-vector recurrent neural network (MV-RNN) to encode the input token sequence according to its syntactic parse tree. Various works [42, 43, 18] using CNN with different filter size, shortest-dependency path were proposed for relation classification. The other set of approach includes using LSTMs and BiLSTM [44–46, 7] along dependency path, attention layers, and position-aware encodings.

More recently, graph convolutional neural networks (GCN) for relation extraction gained a lot of interest. For example, Zhang et al. [47] apply graph convolutions over the dependency tree of input sentence along the shortest dependency path. Similarly, self-attention models such as transformer has also received attention recently. For instance, [48] extend the transformer architecture for relation classification. Some of the works following BERT based architecture [9, 49–51] have established state-of-the-art result on different RC datasets.

Different deep neural models have different characteristics. Thus, they have provided various advantages in dealing with various relation classification problems. They are scal-

able and does not require extensive feature engineering. Therefore, they have been highly successful in most of the learning problems.

Chapter 3

Taxonomical Relation Hierarchy and it's application in Relation Classification

Chapter Highlights

- Most of the datasets available for the Relation Classification task have very few relation labels, ignore relation between relations, and are noisy and imbalanced.
- We propose a manual framework to create a taxonomical relation hierarchy from multiple knowledge bases.
- We proposed taxonomical relation hierarchy based filtering heuristic to remove noisy instances and relabeling heuristic to tackle long-tail distribution and overlapping relation boundary problem.
- We propose scaling cross-entropy loss using shortest-path distance between actual and predicted label in relation hierarchy.
- The proposed heuristics improves the baseline model performance on the RC dataset TACRED.
- We also analyzed the model's predictions based on distance between actual label and predicted label to show effectiveness of our proposed loss function.
- Publications related to this chapter are
  - *"Taxonomical Hierarchy of Canonicalized Relations from Multiple Knowledge Bases"* presented at CoDS-COMAD 2020.
  - *"Improving Relation Classification Using Relation Hierarchy"* presented at NLDB-2022.

Abstract

Due to the availability of large datasets, deep neural models have achieved remarkable success in numerous domains. However, the quality of labeled data is a concern since the majority of the large crowd-sourced dataset generation relies on untrained annotators and imbalanced corpora. Problems such as long-tail distribution, ambiguous label boundaries, and noisy labels severely impact the generalization capabilities of deep learning models. Thus, taking care of the above-mentioned challenges is one of the open areas in the RC domain. The work in this chapter is based on the assumption that exploitation of relation among relations can help in taking care of these challenges to some extent.

In this chapter, our objective is to address these challenges of the RC task using a novel relation hierarchy. Towards this objective, we address two important questions. *First*, how do we define an unambiguous and interpretable taxonomical (`is-a`) relation hierarchy? *Second*, how do we utilize the relation hierarchy for improving RC datasets and models? To answer the first question, we obtain a representative large number of relations from knowledge bases, focusing on *person*, *organization* and *location* entity types. Further, we canonicalize, filter, and combine the identified relations from the previous step to construct a taxonomical relation hierarchy (TRH) of 623 canonical relations.

Finally, for the second question, we explore the widely used large RC dataset, TACRED, to identify the potential impact of using relation hierarchy for RC. We first arrange the relation labels of TACRED into a relation hierarchy, following the template of TRH. This arrangement led us to the following observations: (i) the object entity types of some instances do not align with the annotated relations; (ii) the existence of overlapping relation label boundaries; and (iii) the long-tail distribution of relation labels. Subsequently, we propose TRH-based filtering and relabeling heuristics to address these challenges. Lastly, we propose a scaled cross-entropy loss using the shortest-path distance between ground truth and the predicted label from the relation hierarchy. Our extensive empirical analyses indicate filtering and relabeling, and the scaled cross-entropy loss help improve the model performance. These improvements show the positive impact of using relation hierarchy for the RC task.

3.1 Introduction

In recent years, relation classification (RC) has gained significant attention in the NLP research community. The surge in attention is due to RC being an important intermediate

step for several NLP tasks, including Automatic Knowledge Base Completion (AKBC) [52], Information Retrieval (IR) [53], Reasoning [54], and Question Answering (QA)[55]. However, the adaptation of RC in the pipeline for the mentioned downstream NLP task has been frustratingly slow because of the average performance of deep neural models on the large RE/RC datasets [7, 29]. The performance of a deep neural model largely depends on the training dataset. And our studies have shown that the existing RC resources [27, 26, 29, 7] have at least one of the following bottlenecks: *limited number of relation labels*, *absence of canonical relations*, and *noisy and imbalanced datasets*.

The first and second limitations are because of the pre-defined handcrafted or corpus-dependent relations list [27, 26]. To scale the number of relations, a few datasets [29, 56] rely on external knowledge bases using distant supervision strategies. However, the generated dataset is noisy due to incomplete knowledge bases. Creating a large manually annotated dataset necessitates the involvement of a subject expert, making the process time- and cost-intensive. To mitigate this high labeling cost, researchers rely on non-expert platforms, such as Amazon's Mechanical Turk (AMT). However, the use of such platforms results in unreliable labels due to inexperienced crowd annotators, as in the case of TACRED [7], a large crowd-sourced RC dataset.

To address the challenges mentioned above, we conducted a thorough study of relational resources. And our preliminary study suggests that, *(i)* even though KBs like Wikidata [2] and DBpedia [1] incorporate deep hierarchical ontologies, they do not explicitly address relations, and the unambiguous extraction of relation hierarchy from them is non-trivial. *(ii)* several incorrect relation labeling can be identified by examining the corresponding $subject\_entity$ and $object\_entity$. For example, relation `PARENT` exists for both *organization* and *person*, but an *organization* cannot be `PARENT` to *person* and vice-versa.

Thus, it is important to create a large database of relations, that considers *relation* as a concept and is defined based on $subject\_entity\_type$ and $object\_entity\_type$. Further, it must cover all possible relations that could exist between a pair of entities, taxonomic and semantic associations between relations, and their synsets. This study initiates work in that direction. We assume properties and attributes appearing in structured knowledge bases (KBs) and ontologies are good representatives of all possible relations. Therefore, we extract relations from Wikipedia Infobox templates[1] manually, and from DBpedia and Wikidata triples in a data-driven way. We collect an exhaustive list of relations between *PERSON*, *ORGANIZATION*, and *LOCATION* entity types. Following that, we construct a taxonomical relation hierarchy (TRH) of 623 canonical relations.

[1] https://en.wikipedia.org/wiki/Wikipedia:List_of_infoboxes

Further, to explore the use cases of TRH in learning from noisy and imbalanced RC datasets, we thoroughly analyze the RC dataset TACRED [7] and the state-of-the-art RC models[7, 57, 9, 51, 50]. Our analysis indicates the following challenges: *(i)* the presence of instances in which subject or object entity types do not match with the annotated relation label, *(ii)* the presence of long-tail relation classes, and *(iii)* the presence of a few relations with overlapping definition.

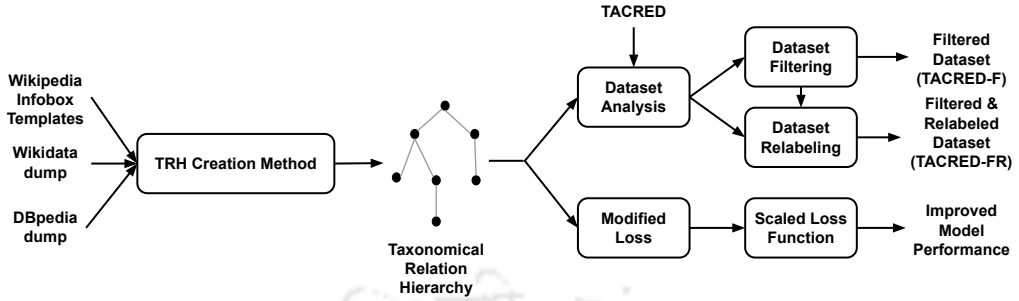
Using TRH as a template, we first create a relation hierarchy for TACRED relation labels. Then, we introduce a filtering heuristic to eliminate instances with ambiguous *subject-object-entity-type-pair* from the TACRED, generating a variant dataset *TACRED-F* for the above-mentioned first challenges. Next, we relabel instances associated with similar relational concepts with coarser relation labels following the relation hierarchy (*TACRED-FR*) to address the second and third challenges. Our heuristics have shown improved performance across all the baseline models.

Finally, we have used the hierarchical distance between relation labels in the relation hierarchy to scale the cross-entropy loss (CE Loss). The primary objective here is to penalize more if the distance between the ground-truth and the prediction in the relation hierarchy is large. For illustration, let us consider an example sentence: *"Apple was founded by Steve Jobs"*. The ground-truth relation is $ORG:FOUNDER$ between the entities `Apple` and `Steve Jobs`. If a model M1 predicts $ORG:TOP\_MEMBER/EMPLOYEE$ and model M2 predicts $PER:SCHOOL\_ATTENDED$, then the prediction from M2 will be penalized more than the prediction from M1. The proposed *Hierarchical Distance Scaled Cross-Entropy Loss (HCE Loss)*, scales the loss value by 2 and 6 for M1 and M2, respectively. The empirical results using SpanBERT[9] show that model perform better with HCE loss compared to CE loss. Further analysis of the results of models using HCE loss shows that the number of misclassifications, where the distance between predicted and ground-truth labels is high, is reduced significantly.

Figure 3.1 provides an overview of the work. The primary focus of this work is to show a few examples of effective use of relation hierarchy for relation classification from noisy and imbalanced datasets. In this process, this chapter makes the following major contributions:

- Create a taxonomical relation hierarchy of more than 600 relation labels using multiple knowledge bases.
- TRH based filtering and relabeling heuristics to target RE/RC dataset challenges such as, incorrect object entity type sentence detection, long-tail distribution, and overlapping relation label boundary.
- Using hierarchical distance to scale cross-entropy loss for better optimisation.

Fig. 3.1 Overview of our work.



3.2 Literature Review

3.2.1 RE Datasets

The majority of the earlier RC models have used ACE multilingual datasets [27] and SemEval-2010 task 8 [26]. The most commonly used ACE 2004 dataset has 16,771 annotated instances labeled with 24 fine-grained relation labels. Whereas the SemEval dataset has 10,717 instances annotated with 19 asymmetric semantic relations (Table 3.1). Given their size, the number of relation labels, and the type of relation labels, these datasets are not useful for large-scale relation classification tasks. Thus, to overcome these challenges, researchers have adapted *Distant Supervision (DS)* startegies[28, 29]. DS has provided a way to address the absence of a large-scale RE/RC dataset at a low cost by leveraging a knowledge base. But, in practice, the generated dataset is highly noisy. Noise is primarily due to the use of a single and incomplete KB, leading to a significantly large number of incorrectly labeled instances. The use of multiple KBs is still not possible as different KBs do not use the same ontology and nomenclature.

TACRED [7], a large crowd-sourced supervised RC dataset, has become popular in recent years. It contains more than 100,000 instances labeled with one of the 41 relation classes or *NO_RELATION* (Table 3.1). However, 80% of the dataset is labeled as *NO_RELATION* and some recent studies [6, 8] have suggested that it contains significant annotation errors. On further evaluation of the dataset, we observed that amongst positive instances there is a significant imbalance indicating long-tail distribution (the number of instances varies from less than 20 to more than 2000 for different relation classes). There are several relation labels, where it is hard to draw hard label boundaries, for example, $PER:COUNTRY\_OF\_BIRTH$ and $COUNTRIES\_OF\_RESIDENCE$, $ORG:SUBSIDIARIES$ and $ORG:MEMBERS$. All these challenges motivate us to use this dataset for our work.

Dataset	# Rel.	# Examples
SemEval-2010 Task 8	19	10,717
ACE 2004	24	16,771
NYT Corpus	52	695,059
TACRED	42	106,624

Table 3.1 A comparison of RC dataset statistics.

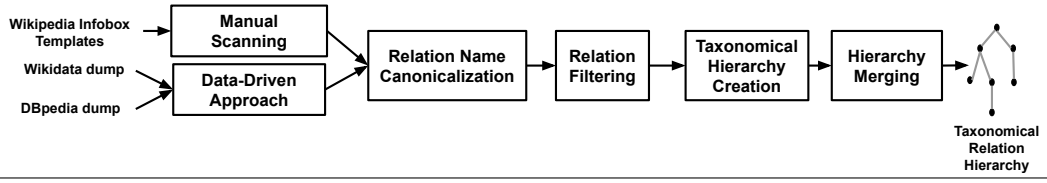
3.2.2 Relation Classification Models and Relation Hierarchy

Earlier methods relied on statistical learning [40, 33, 38, 39] with hand-crafted feature set on ACE datasets [27]. They were later followed by deep neural models [42, 44] on the SemEval [26] dataset. In recent years, TACRED [7] has become the most-widely used dataset for RC because it contains a relatively high number of relation classes and corresponding instances. Various deep learning approaches such as position aware attention on top of LSTM [7], graph convolutional network over dependency parse tree of a sentence [57], transformer [19] based [48], fine-tuning pre-trained BERT model [58], knowledge infused language models [59], and models [50] utilising entity specific information along with language models have been explored.

Given the absence of a hierarchy of relations in RC datasets, very few efforts have been made in this direction. Nonetheless, previous works, such as [60] and [61] aim for hierarchical relation extraction on distantly supervised `2010 NYT dataset` [29] by introducing a hierarchical attention model and reinforcement learning respectively. Earlier, Wang et. al.[62] also proposed a SVM-based model for hierarchical RE on `ACE 2004` dataset. In a recent work [63], Chen et. al. proposed a bi-directional LSTM network along with a loss function to optimize the RC model. They have used the hierarchy from Wikidata. While defining the loss function, they consider the relations of each layer as the ground truth to make a prediction and compute the loss at each layer as the hierarchical loss. Nonetheless, these works are subject to the following restrictions: *(i)* They consider a very shallow hierarchy specific to the given dataset, *(ii)* The use of hierarchy is restricted to their proposed methodology, and *(iii)*. Their proposed approach cannot be generalized to general RC.

Building and utilizing a relation hierarchy is largely an unexplored research direction in the RC domain. In this work, we introduce a relation hierarchy by organizing more than 600 relations between `PERSON`, `ORGANIZATION`, and `LOCATION` entity types collected from three KBs. We further derive a similar relation hierarchy for TACRED and use it to explore the impact of using a relation hierarchy on the performance of baseline models. To the best of

Fig. 3.2 Framework for creating taxonomical relation hierarchy using Wikipedia Infobox Templates, Wikidata, and DBpedia.



our knowledge, this is the first work that utilizes label hierarchy in learning from noisy and imbalanced datasets.

3.3 Relation Hierarchy

A relation triple (e_1,r, e_2) represents relation r between subject entity e_1 and object entity e_2. We extract triples from DBpedia and Wikidata dump. The triples are later used for finding relations and support set of relations.

Figure 3.2 summarizes the following steps of hierarchy creation:

- we start with extracting relations from Wikipedia Infobox templates manually, and DBpedia and Wikidata in a data-driven way between *person*, *organization*, and *location* entity types.
- In step 2 we canonicalize relation names for smooth merging of hierarchy.
- In step 3, we filter relations from the list based on the frequency.
- In step 4, we create hierarchy for individual knowledge resource manually.
- Finally, we create a relation hierarchy of 623 canonical relations.

All the mentioned steps are discussed in details in following subsections,

3.3.1 Getting relation list

Wikipedia infobox stores structured information in the form of attribute-value pairs following an infobox template. Since entries in the infobox are done manually by crowd-sourced workers, we observed lots of irregularities while parsing the infobox. Therefore, instead of collecting triples and relations automatically from infobox, we chose to manually scan

template pages to curate a list of relations. We selected 170, 77 and 89 infobox templates for *person*, *organization*, and *location* respectively. We followed Wikipedia infobox template categories while selecting templates. And used translation count[2] for filtering templates. We refer to this list of relations as $\mathcal{L}_{\text{i}}$. For DBpedia and Wikidata, we follow a data-driven approach. for generating triples, we parse Wikidata JSON dump[3] and DBpedia mapping-based Infobox dump[4].

Then we collect all the unique relations from the triples dataset (where $\texttt{e}_1$ and $\texttt{e}_2$ are one of the three types: *person*, *organization*, and *location*). The two lists of relations are henceforth referred as $\mathcal{L}_{\text{d}}$ (from DBpedia) and $\mathcal{L}_{\text{w}}$ (from Wikidata).

3.3.2 Name canonicalization

Even though the three sources are closely related, they follow different nomenclature for their relations. Thus, the same relation can have different names in different lists. For example, consider relation `placeOfBirth`, which is `birth_place` in Wikipedia Infobox , `birthPlace` in DBpedia, and `place of birth` in Wikidata.

To canonicalize relation names, we follow steps from the Algorithm 1. If a relation name is a single word, consider it as it is, given all the characters are in small-case. Otherwise, capitalize all the words except the first word, remove the in-between spaces, and concatenate all the words. For example, *place of birth* becomes `placeOfBirth`. In the case of multiple names for the same relation (as in the earlier example), we choose one of them and store the respective mapping. Following this procedure, we obtain canonicalized relation lists $\mathcal{C}_{\text{i}}$, $\mathcal{C}_{\text{d}}$ and $\mathcal{C}_{\text{w}}$ from $\mathcal{L}_{\text{i}}$, $\mathcal{L}_{\text{d}}$ and $\mathcal{L}_{\text{w}}$ respectively.

3.3.3 Filtering

This step ensures that our relation hierarchy focuses on frequently occurring relations. We analysed the frequency of each relation label across infoboxes to identify the threshold for filtering. We observed that more than 60% of the relation labels have frequency of more than 100. Further, it covers the proportionate number of relations for all three entity types (Person, Location, and Organisation). This was not the case when the number was 200, 500, or 1000. We adopted to the same threshold (100) for filtering from both DBpedia and Wikidata.

[2]Number of Wikipedia pages that use the template
[3]https://dumps.wikimedia.org/enwiki/
[4]http://downloads.dbpedia.org/2016-10/core/

Algorithm 1 Steps for canonicalising relation names

Input: relation r
Output: $c\_relation$

if $len(relation.split()) == 1$ **then**
  $c\_relation = relation.lower()$.
else
  $num\_words = len(relation)$
  $words = relation.split()$
  while $num\_words > 0$ **do**
    $c\_relation = c\_relation + words[num\_words].capitalize()$.
    $num\_word = num\_word - 1$
  end while
end if
return $relabel\_sids, eliminate\_sids$

We filter out relations from the list $\mathcal{C}_i$ if they appear in less than 100 infoboxes. Similarly, a relation is filtered out from the lists $\mathcal{C}_d$ and $\mathcal{C}_w$ if it has less than 100 associated triples in their support set.

3.3.4 Hierarchy creation

According to [64], a relation `r` describes a relationship between two entities by associating them with certain entity types. Thus, it is natural to classify based on the subject and object entity types at the top level. Consider the relation `founder`, subject entity type is organization (`org`) and object entity is of type person (`per`), thus it falls under the branch `org-per`. Since it is organization specific relation, `org-per.founder` will fall under `org` which falls under the root `rel`.

Initial levels of hierarchy are described as:

- At depth 0: root node referred as `rel`
- At depth 1: we distinguish based on subject entity type. In this level there are 3 nodes `per` (person relations), `loc` (location relations), and `org` (organization relations).
- At depth 2: we distinguish based on both subject and object entities. In this level, there are 9 nodes (For example, `per-loc` subject entity: *person* and object entity: *location*) and each node of this level are henceforth referred as a bucket for relations.

All the relations from the three filtered lists are distributed across 9 buckets. We manually arrange relations in the hierarchy whenever *is-a* association exists between two relations.

Taxonomically, similar relations (child nodes) are placed under the same parent node. If the parent node is not present in the filtered relation list, the canonical relation list is referred to. If present, that referred relation is chosen. Otherwise, a new parent node is introduced. In our hierarchy, we have introduced a total of 12 new nodes.

Hierarchy creation is done manually based on our collective judgment following the relation triples (collected from KBs) associated with each relation. Manual efforts ensure the hierarchy is more interpretable and noise-free.

3.3.5 Hierarchy merging

Hierarchies $\mathcal{H}_{\text{i}}$, $\mathcal{H}_{\text{d}}$ and $\mathcal{H}_{\text{w}}$ are created following the guidelines from the previous step. Finally, they are merged into one common hierarchy $\mathcal{H}$ by eliminating the duplicates and placing taxonomically similar relations under the same branch.

3.4 Application of Relation Hierarchy: Case study on TACRED

We explore some of the potential uses of a relation hierarchy with the help of TACRED, a large-scale noisy and imbalanced relation classification dataset. In particular, we present the following use cases: (i) ambiguous instance filtering, (ii) relabeling of instances belonging to overlapping relation classes, and (iii) scaled cross-entropy loss definition using the relation hierarchy.First, we briefly describe the TACRED and the creation of a hierarchy of TACRED relation labels following the TRH template discussed in Section 3.3.

3.4.1 TACRED and corresponding relation hierarchy

TACRED is a sentence-level dataset where each instance contains a sentence and a pair of entities. One entity represents the *subject entity* and the other represents the *object entity*. Each instance is labeled with one of the 41 relation labels (positive label) or `no_relation` (negative label). Negative instances account for almost 80% of the entire dataset. The

TACRED dataset has a total of seventeen entity types. We arranged these fine entity types into four coarse entity types. Table 3.2 demonstrate the mapping between the entity types. We observe that subject entity types for all relations in TACRED are either *PERSON* or *ORGANIZATION* and object entity types can be mapped to one of the four coarse types, namely, *PERSON*, *ORGANIZATION*, *LOCATION*, and *MISC*.

Coarse Entity Type	Fine Entity Types
PERSON	PERSON
ORGANIZATION	ORGANIZATION
LOCATION	CITY, STATE_OR_PROVINCE, COUNTRY, LOCATION
MISCELLANEOUS	TITLE, RELIGION, NUMBER, URL, DATE, DURATION, NATIONALITY, CAUSE_OF_DEATH, MISC, CRIMINAL_CHARGE, IDEOLOGY

Table 3.2 Mapping between coarse and fine entity types

Following the steps mentioned in the earlier Section 3.3, we create eight relation buckets based on the two subject and four object entity types. Relations in each bucket are manually arranged following ***is-a relation***. We have also introduced a few new relations in the hierarchy to better represent the taxonomic relation between the relations. For example, we have introduced a new relation $PER:LOCATION\_OF\_BIRTH$. This relation is the parent node for the following relations: $PER:CITY\_OF\_BIRTH$, $PER:COUNTRY\_OF\_BIRTH$, and $PER:STATE\_OR\_PROVINCE\_OF\_BIRTH$. In all, we introduce a total of seventeen new relations corresponding to non-leaf nodes in the hierarchy. All relations from the original TACRED dataset are mapped to the leaf nodes in the relation hierarchy. Figure 3.3 shows the TRH corresponding to the TACRED relation labels.

3.4.2 Filtering Ambiguous Instances

A relation is defined between a pair of entities; hence, the domain and range of a relation are defined over entity types. Certain relations can be defined with multiple subject and object entity types. For example, relation *PARENT* can exist between two *PERSON*s as well as between two *ORGANIZATION*s. Thus, some relations can have multiple entity types for subject and object entities. Moreover, in one of the recent works [65], while evaluating various RC models, the authors observed that the models rely heavily on entity type information. Therefore, RC needs to consider unambiguous combinations of subject and object entity types for a given relation. However, some instances in the TACRED dataset do not follow this rule. For example, there are instances in the training datasets (Figure 3.4) with *COUNTRY* and *LOCATION* as the object entity type for relation $ORG:SUBSIDIARIES$.

Fig. 3.3 TACRED Relation Hierarchy

Fig. 3.4 Sample example with incorrect annotation from TACRED.

Hailing the initiative, the Afghan finance minister said that the agreement was part of a process that begun in 2007, when **Afghanistan** first begun its debt relief program under the **International Monetary Fund** (IFM) and the World Bank' Heavily Indebted Poor Countries Initiative (HIPC).

(International Monetary Fund, Afghanistan)

Annotated Relation Label: org:subsidiaries

A country or location cannot be a subsidiary of an organization. This indicates an annotation error.

On examining positive instances from the TACRED training dataset, we found that all the positive instances can be divided into 69 distinct relation triples (subj_ent_type, relation,

obj_ent_type). Out of these 69 distinct triples, there are 11 triples for 6 relations with ambiguous object entity types. Table 3.3 lists all such incorrect triples. The table also specifies the percentage of the relation instances belonging to a given triple category. Consider the relation $ORG:ALTERNATE\_NAMES$. Out of all the instances of this relation in the training dataset, 2.85% instances are of the type where the subject entity is an *ORGANIZATION* and the object entity type is *MISC*. We believe that these data points are potentially noisy. Because an alternate name for an organization should be typed as an organization. Similarly, we believe that the rest of the triples in the table also correspond to potentially noisy instances. Samples corresponding to such triples should be eliminated while training and testing the RC models. We eliminated 164 sentences (Table 3.4) from the TACRED dataset following this approach.

subj_ent_type	relation	obj_ent_type	% instances for a relation		
			train	dev	test
ORGANIZATION	ORG:ALTERNATE_NAMES	MISC	2.85	2.37	0.94
ORGANIZATION	ORG:MEMBER_OF	COUNTRY	13.11	6.45	16.67
ORGANIZATION	ORG:MEMBER_OF	LOCATION	9.02	3.23	5.56
ORGANIZATION	ORG:MEMBER_OF	STATE_OR_PROVINCE	4.1	0	0
ORGANIZATION	ORG:PARENTS	COUNTRY	5.24	4.17	3.23
ORGANIZATION	ORG:PARENTS	LOCATION	2.45	1.04	1.61
ORGANIZATION	ORG:PARENTS	STATE_OR_PROVINCE	1.05	0	0
ORGANIZATION	ORG:SUBSIDIARIES	COUNTRY	0.34	1.77	0
ORGANIZATION	ORG:SUBSIDIARIES	LOCATION	9.12	13.27	2.27
PERSON	PER:ALTERNATE_NAMES	MISC	7.69	2.63	9.09
PERSON	PER:EMPLOYEE_OF	LOCATION	0.07	0.27	0.38

Table 3.3 Statistics of incorrect combinations of subject and object entity types for a given relation

Based on manual evaluation of misclassified instances, [6] has shown that there are significant numbers of instances incorrectly assigned as *NO_RELATION*, even though one of the 41 relations is valid. However, identifying incorrectly negatively labeled instances is beyond the scope of this work.

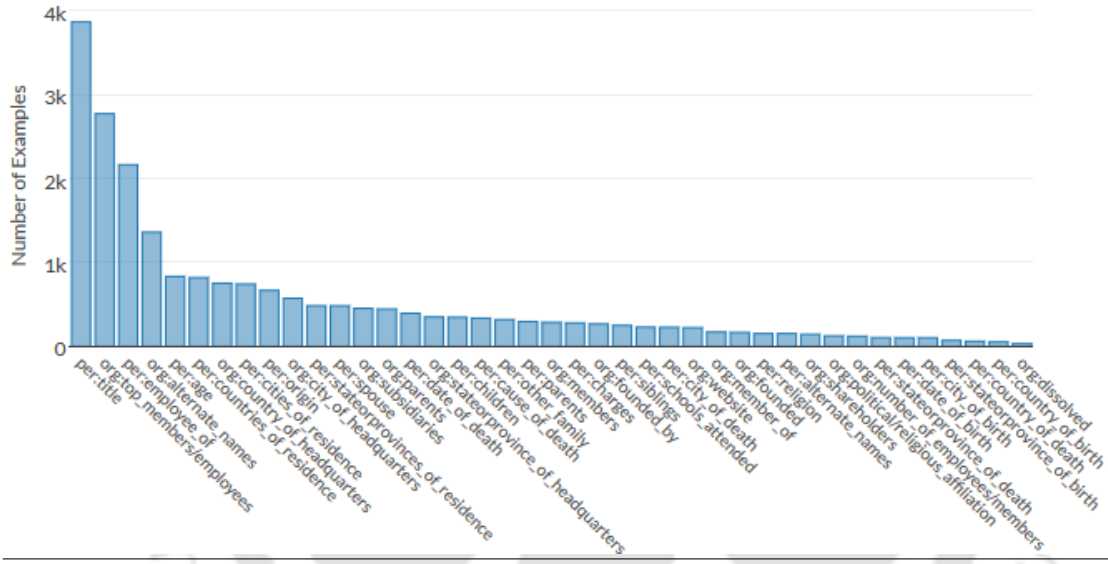
Partition	#examples filtered	% of Positive labeled instances
train	117	0.9 %
dev	35	0.64 %
test	12	0.36 %
total	164	0.75 %

Table 3.4 The Number of instances filtered from each partition of TACRED. Total number of positive train instance: 13012, positive dev instances: 5436, and positive test instances: 3325 .

3.4.3 Fine to Coarse Re-Labeling

In supervised datasets, data sparsity for some class labels is a very common problem. Furthermore, this problem leads to the long tail distribution of class labels, eventually hurting the overall model performance. TACRED also has an inequitable distribution of instances in training data. For example, Figure 3.5 shows that each of the top 3 relations has more than 2000 instances, while the bottom 5 have fewer than 100 instances.

Fig. 3.5 Histogram representing distribution of examples in training data across relation labels. Source: https://nlp.stanford.edu/projects/tacred/



The other challenge with RC datasets is the identification of class boundaries between certain relations labels due to overlapping contexts. This problem is presented for relations *LOCATION_OF_BIRTH*, *LOCATION_OF_DEATH* and *LOCATIONS_OF_RESIDENCES* in Table 3.5. When one such relation is also part of the long-tail distribution, then learning about such relation labels gets even more difficult. For example, relation *COUNTRY_OF_DEATH* has only 5 training instances, whereas the relation *COUNTRIES_OF_RESIDENCES* has 382 training instances. As a result, all the *COUNTRY_OF_DEATH test* and *dev* instances are classified as *COUNTRIES_OF_RESIDENCES*.

To tackle the above-mentioned problems, we leverage relation between relations from relation hierarchy. We merge the finer relations from relation hierarchy into coarser relations (Table 3.6). For example, relations *CITY_OF_BIRTH*, *COUNTRY_OF_BIRTH*, and *STATE_OR_PROVINCE_OF_BIRTH* are fine-grained for relation *LOCATION_OF_BIRTH*. Thus instances of all 3 relations are merged into one. Although this decreases the granularity

Sentences	Entity Pair	Relation	Other Valid Relation
Steve Jobs was born and raised in California, US, where he later passed away aged 56.	Steve Jobs, California	STATE_OF_RESIDENCE	STATE_OF_BIRTH, STATE_OF_DEATH
Albert Einstein was born in Ulm, Germany.	Albert Einstein, Ulm	CITY_OF_BIRTH	CITY_OF_RESIDENCE
Albert attended Catholic Elementary School in Munich.	Albert, Munich	CITY_OF_RESIDENCE	-
Einstein accepted an earlier offer from the Institute for Advanced Study, in Princeton, New Jersey, to become a resident scholar.	Einstein, New Jersey	CITY_OF_RESIDENCE	-
Albert Einstein died aged 76 in Princeton, New Jersey, US.	Albert Einstein, New Jersey	CITY_OF_DEATH	CITY_OF_RESIDENCE

Table 3.5 Examples illustrating certain relations such as LOCATION_OF_BIRTH, LOCATIONS_OF_RESIDENCES, and LOCATION_OF_DEATH can be derived from a similar context.

it helps in increasing the number of instances for a similar relation concept, thus helping the models learn relation boundaries efficiently. Once the coarser relation is learned, a finer relation label can be learned by modifying the objective as multi-label classification [66] or using hierarchy aware models [67].

Old Relation	Count			New Relation	Count		
	Train	Dev	Test		Train	Dev	Test
ORG:CITY_OF_HEADQUARTERS	382	109	82	ORG:LOCATION_OF_HEADQUARTERS	1079 (1.6%)	356 (1.6%)	241 (1.6%)
ORG:COUNTRY_OF_HEADQUARTERS	468	177	108				
ORG:STATEORPROVINCE_OF_HEADQUARTERS	229	70	51				
PER:CITY_OF_BIRTH	65	33	5	PER:LOCATION_OF_BIRTH	130 (0.2 %)	77 (0.3%)	17 (0.1%)
PER:COUNTRY_OF_BIRTH	27	18	4				
PER:STATEORPROVINCE_OF_BIRTH	38	26	8				
PER:CITY_OF_DEATH	81	118	28	PER:LOCATION_OF_DEATH	135 (0.2%)	202 (0.9%)	41 (0.3%)
PER:COUNTRY_OF_DEATH	5	43	9				
PER:STATEORPROVINCE_OF_DEATH	49	41	14				
PER:CITIES_OF_RESIDENCES	374	179	189	PER:LOCATIONS_OF_RESIDENCES	1087 (1.6%)	448 (2%)	405 (2.6%)
PER:COUNTRIES_OF_RESIDENCES	382	197	135				
PER:STATESORPROVINCES_OF_RESIDENCES	331	72	81				
PER:CHILDREN	211	99	37	PER:FAMILY	965 (1.4%)	424 (1.9%)	306 (2%)
PER:OTHER_FAMILY	179	80	60				
PER:PARENTS	152	56	88				
PER:SIBLINGS	165	30	55				
PER:SPOUSE	258	159	66				

Table 3.6 Relations with the number of instances in TACRED train, dev, and test datasets before and after merging following relation hierarchy. Total number of instances in train:68124, dev:22631, and test:15509

3.4.4 Hierarchical Distance Scaled Cross-Entropy Loss

Cross-Entropy Loss estimates the negative log-likelihood of a class label t_i under a categorical distribution y (Equation (1)).

$$J = -\sum_{i=1}^{n_r} t_i log(y_i) \tag{1}$$

Here, n_r is number of relation labels, $t \in \mathbb{R}^{n_r}$ is one-hot encoding to represent the ground-truth, and $y \in \mathbb{R}^{n_r}$ is the estimated probability for each relation label by softmax classifier.

CE loss does provide a good estimate of how far model is off in predicting the correct label. But how far is prediction from the ground-truth is not efficiently captured. Moreover, [68] have shown that cross-entropy loss function is not robust for multi-class classification with noisy labels.

We introduce hierarchical scaling of cross-entropy loss in this work, considering class labels can be organized in a label hierarchy. We refer to the proposed loss as a Hierarchical distance scaled CE (HCE) loss. We modify the CE loss function by scaling it with the hierarchical distance between the ground-truth and predicted label $d_{y^*,\hat{y}}$ as follows:

$$J^* = d_{y^*,\hat{y}} * J \tag{2}$$

where, J is cross-entropy loss calculated as in Equation (1), y^* is the ground-truth and $\hat{y}$ is the predicted relation label.

This modification ensures that when the distance between the ground-truth and predicted label is high in the hierarchical tree, it is penalized higher compared to when the distance is smaller. For example, in the case of relation classification, if the actual relation between a pair of entities mentioned in a sentence is $PER : CITIES\_OF\_RESIDENCES$ and the predicted relation is $ORG : CITY\_OF\_HEADQUARTERS$ the loss value is multiplied by a higher constant than when the predicted label is $PER : CITY\_OF\_BIRTH$ following the relation hierarchy.

As the HCE loss function involves the multiplication of constants to the CE loss function, the gradient is propagated backward by scaling it with the same factor.

3.5 Experiment Setup

In this section, we discuss empirical setup along with baselines and evaluation metrics for our proposed strategies *filtering*, *relabeling*, and *HCE Loss* on large-scale RC dataset TACRED.

3.5.1 Baseline Models for Dataset Evaluaion

We have used some the state-of-the-art RC model for the dataset evaluation. Brief description of each of them is as follows:

PALSTM [7]: To efficiently encode the contribution of each word and the position of entities in the sentence representation, authors have included position-aware attention on top of the neural sequence model (LSTM [15]). A simple LSTM produces a sequence of hidden states $\{h_t\}_{t=1...T}$, where T is the length of input sequence. The final representation is obtained by taking the dot product of the attention weights a_t with h_t, which is then fed to the fully connected layer followed by a softmax layer for relation classification.

SpanBERT [9]: SpanBERT extends **BERT** model [24] by pre-training with masking a contiguous random span and training the span boundary representation to predict the entire content of the marked span. For RC, they consider the subject and object entity as two spans and predict the relation between them. In the final layer, they add a *softmax* layer on the $[CLS]$ token to predict the relation label. We have used `SpanBERT-base-cased` model for all our experiments.

BERT and RoBERTA [50]: We follow [50] to obtain baseline results of **BERT [24]** and **RoBERTA [25]** models for the RC task. The authors in [50] highlighted the entity information such as *mention* and *type* using special markers and used the concatenation of entity representation for relation classification. Their approach using RoBERTA achieved the best performance on TACRED (F1: 74.2) and ReTACRED (F1: 91.3) beating all other baselines in our re-implementation. We have used `BERT-base-cased` and `RoBERTA-large-cased` models for all our experiments.

We trained all our TACRED-based models using the hyper-parameters reported in their respective contribution.

3.5.2 Experimental Setup and Baseline for Proposed Loss Function

The CE loss function is the most commonly used loss function for multi-class classification problems. Hence, we evaluate the performance of our proposed HCE Loss by considering CE loss as the baseline. For all our HCE loss experimentation, we fine-tuned the hyper-parameters of the SOTA model SpanBERT[9]. With relevant fine-tuning, similar results can be obtained for other models as well.

Since *NO_RELATION* is not part of the relation set, there must be an appropriate penalty associated with it. Thus, we did some experiments to determine the penalty to impose when the true label is a *RELATION* and the prediction is *NO_RELATION* and vice-versa. We observe the best performance corresponds to the case when we penalize the model with twice the maximum distance if the label is a relation and the prediction is *NO_RELATION* and 1.75 times the maximum distance for the reverse misclassification.

3.5.3 Evaluation Metric

For all our experiments, we consider macro-averaged **Precision (P)**, **Recall (R)**, and **F1-Score (F1)**. We compute the metric independently for each relation class r and then take the average. P, R, and F1 are defined by the equations Eq 6, Eq 7 and Eq 8 respectively.

$$Precision(P) = \frac{Number\ of\ instances\ correctly\ classified\ as\ relation\ r}{Total\ number\ of\ instances\ predicted\ as\ relation\ r} \tag{6}$$

$$Recall(R) = \frac{Number\ of\ instances\ correctly\ classified\ as\ relation\ r}{Actual\ number\ of\ instances\ with\ relation\ r} \tag{7}$$

$$F1\ Score(F1) = \frac{2PR}{P+R} \tag{8}$$

We also consider positive accuracy, i.e. how efficient a model is at correctly classifying positive relations for evaluation. This is equivalent to a macro-averaged recall.

All the above-mentioned metrics efficiently provide overall information on how well the model is performing. Even though these metrics are efficient at judging the correctness of predictions made by the RC models, they fail to provide any information on the severity of error while evaluating the RC models. Hence, we have proposed *prediction at distance d* to measure the performance of the model with fine-grained details.

Prediction at distance d shows the fine-grained analysis of model errors. For each test data instance, we will have the Annotation Label (*AL*) from the dataset and the Predicted Label (*PL*) from the model. We locate both *AL* and *PL* in our relation hierarchy and compute the shortest distance path between them. The greater the distance between *PL* and *AL*, the more severe the error in model prediction.

	Relation Count	Relation @d = 3	Relation @d = 4	Relation @d = 5
$\mathcal{H}$	623	357	247	19
$\mathcal{H}_i$	351	177	168	6
$\mathcal{H}_d$	282	162	110	10
$\mathcal{H}_w$	267	209	52	6

Table 3.7 Relation Count of hierarchies and number of relations at various depths (Relation @ d =).

	Person		Organization		Location	
	B	A	B	A	B	A
Infobox	660	154	228	165	183	84
Dbpedia	94	92	103	99	91	86
Wikidata	71	71	73	72	98	97

Table 3.8 Relation counts before (B) and after (A) canonicalization.

3.6 Results & Discussion

3.6.1 Relation hierarchy

The basic statistics of three individual hierarchies $\mathcal{H}_i$, $\mathcal{H}_d$, $\mathcal{H}_w$, and the common hierarchy $\mathcal{H}$ are presented in Table 3.7. All hierarchies have a maximum depth of 5 (6 levels). All relations from the filtered lists are distributed at depths of 3, 4, and 5. The distribution of relations at depths 4 and 5 gives more fine-grained information about relations shared between two entities. In the common hierarchy, `loc-loc` bucket has the highest number of relations (113), whereas `org-loc` bucket has the least number (24).

Effects of canonicalization: Relation name canonicalization has played an important role in eliminating redundant relations from $\mathcal{L}_i$ (Table 3.8). This, in turn, helped in finding common relations among the resources. Since DBpedia and Wikidata are structured at their core, canonicalization has not affected much.

Complementarity of resources: The contribution of resources to relation buckets is depicted in Figure 3.7. Manually collected relations from Wikipedia Infobox dominate 7 out of the 9 buckets. The contributions of DBpedia and Wikidata towards each bucket is almost similar. Figure 3.6 shows the contribution of each resource towards the common hierarchy.

Fig. 3.6 Contribution of resources towards common hierarchy.

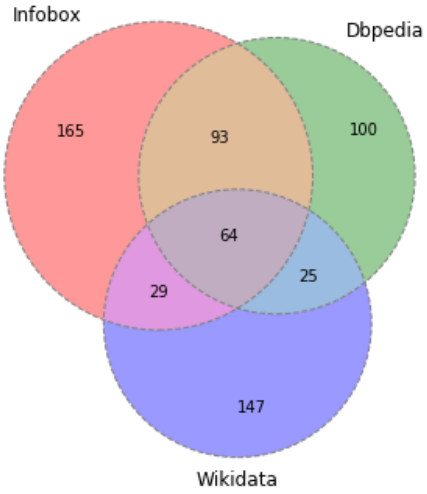
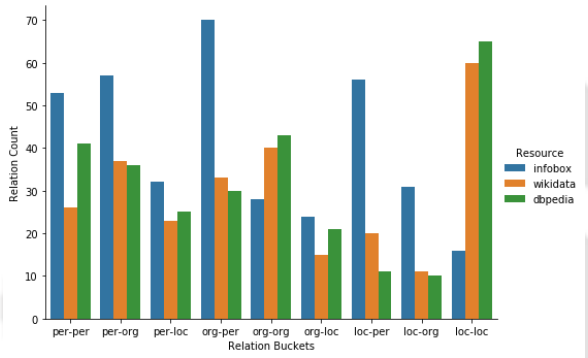


Fig. 3.7 Distribution of relations from different resources in each of the 9 buckets.



Only about 10% relations are common among the three resources. This analysis indicates the complementarity of the resources.

Comparison with relation list of RC datasets: The main objective behind this study was to highlight the major bottlenecks in RC datasets (Sec 3.1). Table 3.9 briefly shows how relations from RC datasets get subsumed in our relation hierarchy. Our hierarchy covers an average of 62% of relations when all the relations in a dataset are considered and 85.35% of relations when the relation's subject and object entity types are restricted to *person*, *organization*, and *location* types.

RC Dataset	relation count	depth	relation subsumed
ACE 2004	24 (17)	1	11
NYT2010 Dataset	51 (47)	0	35
TACRED	41 (29)	0	27
Relation Hierarchy	623	5 (3.42)	-

Table 3.9 RE Datasets with relation count(relation with subject and object entity of type *person*, *organization*, and *location*), hierarchy depth, and numbers of relation subsumed in Relation Hierarchy

3.6.2 Model Performance on TACRED and variants

3.6.2.1 Effects of Filtering:

TACRED is a large crowd-sourced dataset containing more than 100 thousand sentences in total. And like any other large dataset, it is no exception to label noise. Recent studies [6, 8] have shed some light on the noisiness of the TACRED. On top of that, they have also cleaned TACRED by reannotation and have shown performance improvement in their respective new variants. In this work, we have proposed a heuristic utilizing relation hierarchy to clean the noisy positive instances from the dataset. We generated **TACRED-F** by filtering out instances where subject and object entity types do not align with the relation label. Table 3.10 presents the results of our filtering strategy. Almost all the models have shown improved performance, although not very significant since the number of filtered out instances is relatively small. This heuristic can be further improved by considering fine-grained entity type information, but that part is left for future experimentation as of now.

3.6.2.2 Effects of Relabeling

The main motivation behind relabeling instances with their coarser labels is to mitigate the effects of relation labels with fewer instances sharing context with other relation labels. As previously discussed, instances of relation labels in the *per-loc* bucket are similar to one another. For example, it is often the case that $PER : CITY\_OF\_BIRTH$ or $PER : CITY\_OF\_DEATH$ is often the same as $PER : CITIES\_OF\_RESIDENCES$. And relation labels with fewer instances often get classified with the other relation labels. Combining relation labels with similar information improves the training set for individual relation labels.

Model	Metric	TACRED		TACRev	
		Original	TACRED-F	Original	TACRev-F
PARNN[7]	Precision	66.5	63.8	71.1	68.5
	Recall	65.7	68.6	74.7	78.2
	F1	66.1	66.1	72.9	73
SpanBERT[9]	Precision	66.4	67.8	71.3	72.7
	Recall	66.1	65.7	65.6	74.8
	F1	66.3	66.7	73.4	73.8
BERT[50]	Precision	71.1	72.6	75.9	78.1
	Recall	71.4	70.8	81.1	80.9
	F1	71.2	71.7	78.4	79.5
RoBERTA[50]	Precision	75.4	75.3	81.4	81.4
	Recall	73.1	74.3	84.1	85.4
	F1	74.2	74.8	82.7	83.4

Table 3.10 Evaluation results of dataset filtering on TACRED and TACRev following four baseline models.

The table shows the performance of the model when trained and evaluated on the new dataset TACRED-FR. It can be seen that all the models show decent performance improvement.

Alt et al. [6] have discussed in their paper that TACRED relations under the bucket $per-loc$ and $per-per$ are affecting the model performance. We observed performance improvement in all coarser relations except $PER:LOCATIONS\_OF\_RESIDENCES$ after relabeling those instances with their coarser relation labels (Table 3.12).

3.6.3 Performance of Proposed Loss Function on TACRED

To better understand the learning with HCE loss, we tracked the F1-score and accuracy of positive instances on the validation set during the training (Figure 3.8). Even though the model using HCE loss is relatively slow to converge, the model gets better at predicting the positive instances (Figure 3.8b). This shows that scaling with the distance between prediction and ground truth improves learning.

One of the main objectives behind scaling the CE loss using RH was to reduce the number of predictions at larger distances. For example, instances with *AL* belonging to *per* relation bucket getting *PL* from *org* relation bucket. And, it is evident from *Prediction at distance d* results from Table 3.13 that our proposed heuristic is very close to the objective. All the relations inside $subj-entity$ bucket (*per* or *org*) in the TACRED TRH (Figure 3.3) are

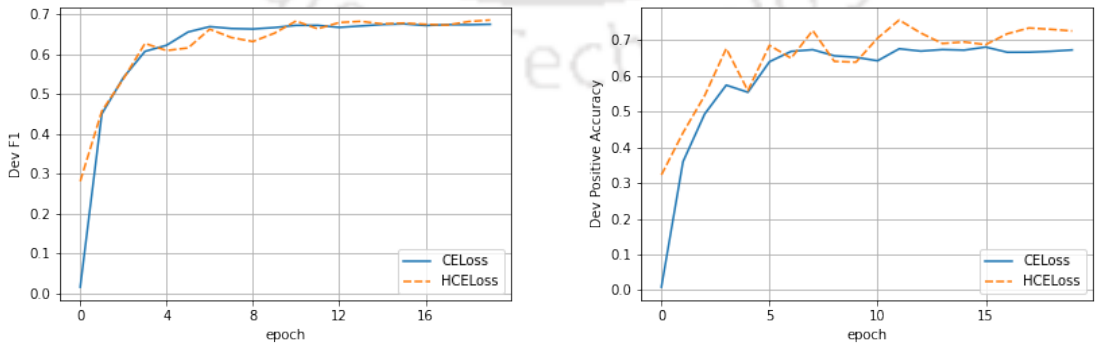
Model	Metric	TACRED		TACRev	
		Original	TACRED-FR	Original	TACRev-FR
PARNN[7]	Precision	66.5	68.1	71.1	73.8
	Recall	65.7	65.7	74.7	75.7
	F1	66.1	66.9	72.9	74.7
SpanBERT[9]	Precision	66.4	70.7	71.3	76.7
	Recall	66.1	65.5	75.6	75.5
	F1	66.3	68	73.4	76.1
BERT[50]	Precision	71.1	73.1	75.9	78.3
	Recall	71.4	71.4	81.1	81.3
	F1	71.2	72.2	78.4	79.8
RoBERTA[50]	Precision	75.4	75.2	81.4	81
	Recall	73.1	74.5	84.1	85.2
	F1	74.2	74.9	82.7	83.1

Table 3.11 Evaluation results of dataset relabeling followed by filtering on TACRED and TACRev following four baseline models.

Relation	#Test Instances	% Correct TACRED Prediction	% Correct TACRED-FR Prediction
PER:FAMILY	306	45.75	72.22
PER:LOCATION_OF_BIRTH	18	27.78	33.33
PER:LOCATION_OF_DEATH	51	15.69	31.37
PER:LOCATIONS_OF_RESIDENCES	418	52.39	51.91

Table 3.12 Comparison of SpanBERT performance for challenging relations group on TACRED and TACRED-FR. *% Correct TACRED Prediction* is the percentage of relation-label instance with correct prediction when trained on TACRED dataset. *% Correct TACRED-FR Prediction* is the percentage of relation-label instance with correct prediction when trained on TACRED-FR dataset.

Fig. 3.8 Dev set performance at different epoch for SpanBERT model.



(a) Dev F1 (b) Dev Positive Accuracy

Dataset	Methods	P	R	F1	Prediction at distance d=x								
					d=0	d=2	d=3	d=4	d=5	d=6	d=7	d=8	d=16
TACRED	CE Loss	66.4	66.1	66.3	2199	53	28	18	43	36	39	5	904
	HCE Loss	68.9	69.2	69.1	2302	89	25	29	3	0	0	0	877
TACRED-FR	CE Loss	70.7	65.5	68	2171	31	0	3	30	25	32	6	1015
	HCE Loss	68.4	68.2	68.3	2259	24	0	0	21	9	31	0	969

Table 3.13 Test Performance of different methods on TACRED dataset using SOTA model SpanBERT. All the results are based on our implementation of original code provided by the author.

at maximum distance of 5, and the entries for distances 6, 7, and 8 in the Table 3.13 are significantly reduced for HCE loss.

Table 3.13 presents the performance of the SpanBERT model using the proposed loss and baseline CE loss functions. The model using HCE loss shows an improvement of 2.8% over the baseline. On further analyzing the prediction made by the model, we observe that *(i)* there is a significant increase in prediction as one of the relation instead of $NO\_RELATION$, *(ii)* earlier some of the instances with relation label $PER:COUNTRIES\_OF\_RESIDENCES$ and $PER:CITIES\_OF\_RESIDENCES$ were predicted as $ORG:COUNTRY\_OF\_HEADQUARTERS$ and $ORG:CITY\_OF\_HEADQUARTERS$ respectively, but with HCE Loss that got reduced.

The qualitative error analysis of the propose loss function, HCELoss, is presented in the Table 3.14. It can be observed from the selected samples that,

1. Examples 1, 3, and 4 show improvement in models' confusion between highly similar contexts.
2. Example 2 shows correction in longer distance prediction.

Both these observation supports our intuition behind using the hierarchical distance between class labels for scaling the cross entropy loss function.

3.7 Chapter Summary

This work builds upon the creation of a taxonomical relation hierarchy (TRH) of 623 relation labels, exploring more than 1500 properties and attributes from multiple knowledge bases, and shows a few important impacts on the large-scale relation classification task. It first discusses how flat labels of any existing relation classification dataset can be converted into a

Sentence	Gold Label	CELoss Prediction	HCELoss Prediction
Although her family was from Arkansas, she was born in Washington state, where her father was working on a construction project.	per:stateorprovince_of_birth	per:stateorprovinces_of_residence	per:stateorprovince_of_birth
" I learn from students and I challenge them," says Heloise, 58, who took over the family hints business when her mother, also named Heloise, died in 1977.	per:parents	per:other_family	per:parents
Wen's wife Zhou Xiaoya and three senior Chongqing police officers - Huang Daiqiang, Zhao Liming, and Chen Tao - will also receive their verdicts at the same court Wednesday afternoon .	per:cities_of_residence	org:city_of_headquarters	per:cities_of_residence
American International Group said it had transferred ownership to the Federal Reserve Bank of New York parts of two international subsidiaries: American Life Insurance Company (ALICO) and American International Assurance Company (AIA).	org:parents	org:subsidiaries	org:parents

Table 3.14 Qualitative Analysis of sample test instances for Hierarchical Loss Function. The subject entity is highlighted in red color and the object entity in green color.

TRH. Second, it shows the use of TRH in efficient learning from the noisy and imbalanced RC dataset TACRED by filtering and relabeling the challenging instances. And, finally, this work shows the use of scaled CE loss using the TRH can further improve the performance of the models.

Our proposed approach could improve the model's performance by 1% to 3%, whereas there are few works involving experts [6] and crowd-sourced reannotation [8] improving the performance by 8% to 13%. This implies that there are still challenges in TACRED that need to be explored. Furthermore, because our proposed heuristics are built upon the basic relation definition of *subject_entity* and *object_enity*, it is difficult to address any issues associated with *NO_RELATION* instances, which alt et al.[6] have shown to be noisy.

This highlights the need for a low-cost, thorough evaluation of both positive and negative instances. We will work in this direction in Chapter 4. We have also discussed some of the future works at the end in the Chapter 6.

Chapter 4

Model-based Characterization and Reduction of noisy instances from RC dataset

Chapter Highlights

- Existing Works have shown that TACARED is a noisy dataset.
- In our previous chapter, we have proposed relation hierarchy based heuristic to filter-out the noisy instances. However, we could only filter a few positive instances.
- The underlying reason for the noise in TACRED is still largely unexplored.
- In this chapter, we first propose a model-based characterization of noise in the TACRED dataset.
- We thoroughly analyzed baseline models' prediction and performance.
- Our analysis indicates that the negative examples, i.e. examples labeled with *NO_RELATION* are the main source of noise in the dataset.
- We explore two semi-automated nearest-neighbour strategies to reduce noise from the TACRED by elimination and reannotation.
- The first strategy, *Intrinsic Strategy (IS)* is based on finding the noisy nearest neighbours of instances with false negative predictions.
- The second strategy, *Extrinsic Strategy (ES)* is based on finding the noisy nearest neighbours of a clean subset of instances.

- Our empirical analysis of both the strategies suggests that elimination and reannotation following ES provide better results.
- This chapter is based on the paper "Noise in Relation Classification Dataset TACRED: Characterization and Reduction" submitted to the ACM Transactions on Knowledge Discovery from Data (ACM TKDD).

Abstract

In the previous chapter, we addressed the first four RC challenges, highlighted in Chapter 1, using taxonomical relation hierarchy-based heuristics. Our empirical observations and earlier studies suggest that label noise in the TACRED is far more prominent than any other challenge, considering the improvement in models' performance when reannotating or filtering some of these noisy instances. All these approaches entirely focus on finding noisy instances by following experts, heuristics, or crowd efforts. Consequently, the nature of noise in the dataset remains largely unexplored. Therefore, in this chapter, we *first*, explore model-based approaches to characterize the primary cause of the noise in TACRED. *Then*, we identify the potentially noisy instances.

To achieve the first goal, we examine the predictions and performance of state-of-the-art (SOTA) models to identify the source of the noise in the dataset. Our analysis of TACRED shows that the majority of the noise in the dataset originates from the instances labelled as no_relation which are negative examples. For the second objective, we explore two nearest-neighbor-based strategies to automatically identify potentially noisy examples for elimination and reannotation. Our first strategy, referred to as *Intrinsic Strategy (IS)*, is based on the assumption that positive examples are clean. Thus, we have used false-negative predictions to identify noisy negative examples. Whereas, our second approach, referred to as *Extrinsic Strategy*, is based on using a clean subset of the dataset to identify potentially noisy negative examples.

Finally, we retrained the SOTA models on the eliminated and reannotated datasets. Our empirical results based on two SOTA models trained on TACRED-E following the *IS* show an average 4% F1-score improvement, whereas reannotation (TACRED-R) does not improve the original results. However, following *ES*, SOTA models show the average F1-score improvement of 3.8% and 4.4% when trained on respective eliminated (TACRED-EN) and reannotated (TACRED-RN) datasets. We further extended the *ES* for cleaning positive

examples as well, which resulted in an average performance improvement of 5.8% and 5.6% for the eliminated (TACRED-ENP) and reannotated (TACRED-RNP) datasets respectively.

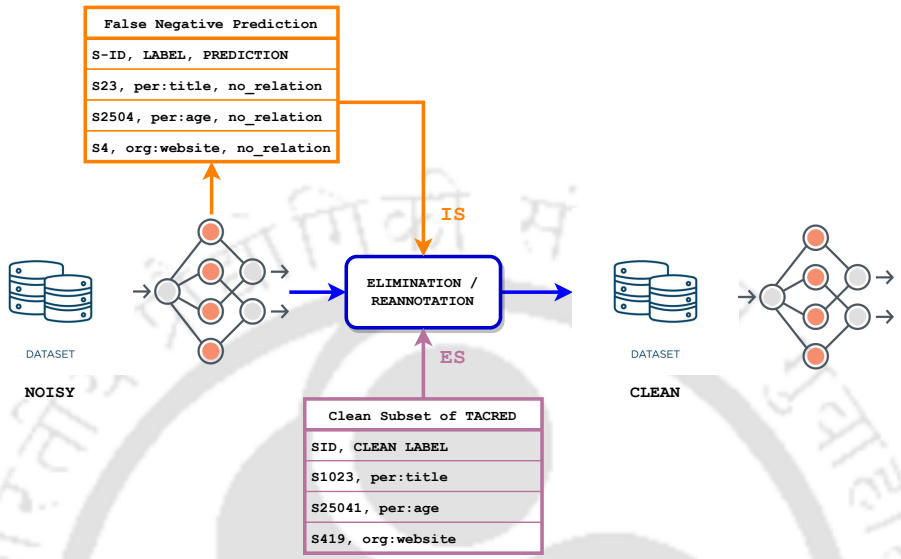
4.1 Introduction

Recent studies [6, 8] have shown that the crowd-sourced TACRED contains significant annotation errors. Both these studies primarily rely on manual intervention in reannotating the partial or complete TACRED. Although [6] did use multiple models to identify the most challenging or error-causing instances from *test* and *dev* sets of TACRED, a subset of the identified instances was manually reannotated. On the other hand, [8] comprehensively analyzed the TACRED instances manually and modified the annotation guidelines for crowd-sourced reannotation. Both studies have shown significant performance improvement on the respective revised test sets. Even though the impact of these studies is significant, primary dependence on manual intervention is a limiting factor due to the associated high cost of time and money. This motivated us to ask a few questions: *(i)* Can we identify the noisy examples at a reasonable cost? *(ii)* Can we perform automated reannotation with minimal human intervention?

Based on the above discussions and with the motivation to answer the above two questions, in the previous chapter, we resorted to relation hierarchy to identify the noisy instances without any human effort. However, despite the small performance gain, the approach was limited to a few noisy positive examples only. Furthermore, the characteristics of noisy instances remained unexamined. Therefore, in this work, we aim at *(i)* systematically studying and identifying noise present in the dataset using RC models only, *(ii)* exploring model-based strategies to identify noisy instances for elimination and reannotation, and *(iii)* performing comparative analyses on these strategies.

To achieve the first objective, we deploy standard exploratory data analysis and machine learning tools to identify the noise present in TACRED. Confusion matrix analysis on the test set to generate a hypothesis on the characteristics of misclassified instances. Positive Relation Classification Analysis to identify the strength of the baseline models. Another set of analyses includes downsampling and binary classification between *relation* and *no_relation* to differentiate between the impact of imbalanced and the noisy nature of data. Lastly, t-SNE plot on a sample of test instances and top-k evaluation to analyze the nature of negative examples in the dataset and models' prediction. All this analysis ensures an automated approach to identifying the cause of noise in the dataset. Moreover, all this analysis can

Fig. 4.1 Pipeline for identifying noisy examples. The two strategies differs in seed set used for identifying noisy examples. Intrinsic Strategy: Orange (highlighted) represents using model's false negative prediction. Extrinsic Strategy: Purple (highlighted) represents using clean subset of TACRED.



be done for any classification task, thus making our approach generalized for any task or domain.

Once the impact of the noisy nature of data on the model performance is established, the next obvious questions are *what to do and how to improve the dataset quality*. "Elimination and "reannotation" of noisy instances are two possible answers to the *what question*. For the *how* question, we chose a model-based approach over manual intervention primarily due to its cost-, time-, labor-effectiveness, and generalizability across datasets and tasks. We explore two different strategies using the nearest-neighbor-based approach to automatically identify potential noisy examples for elimination and reannotation. Figure 4.1 illustrates the general framework adopted in this study.

The first strategy, *Intrinsic Strategy (IS)*, assumes that the instances with positive relation labels are clean and noise is present in instances labeled with *no_relation*, i.e., negative labeled instances are potentially noisy. The assumption is based on the conclusion of the above-mentioned analyses. With this assumption, the first strategy uses positive labeled instances misclassified to *no_relation* class as a seed set to identify noisy negative examples. The second strategy, *Extrinsic Strategy (ES)*, uses a clean subset of TACRED instances to identify noisy samples. In the absence of a budget for manual reannotation, we assume the ReTACRED test set is clean. A subset of it is chosen as clean data to identify noisy instances in the TACRED.

Extensive experimental analysis is done using two basic baseline models PALSTM [7] and CGCN [57]. These two models are chosen mainly due to the following reasons: *(i)* If relatively simple, yet competitive models can show performance improvement, then complex models are more likely to show improvement, and *(ii)* limited computing resources are required for these models. Empirical results show that eliminating or reannotating a small set of examples using a subset of a clean dataset can significantly improve the process of finding noisy instances. Our proposed strategy ES has shown an impressive average improvement of 3.8% and 4.4% when trained on eliminated (TACRED-EN) and reannotated (TACRED-RN) datasets respectively and an average improvement of 5.8% and 5.6% for the eliminated (TACRED-ENP) and reannotated (TACRED-RNP) datasets respectively. Furthermore, manual analysis of 100 instances from TACRED-EN and TACRED-RN shows that the error rate is around 15% only.

Contribution of this chapter can be summarized as follows:

1. Automatic or model-based characterization of noise in the TACRED dataset.
2. Exploring automated elimination and reannotation strategies of a noisy dataset.

The rest of the chapter is organized as follows: in Section 4.2 we have discussed the related works under two sub-sections: dataset evaluation and analysis and dataset reannotation. In Section 4.3, we will show our analysis of the baseline models' performance on TACRED to identify the cause of the noise in the dataset. In Section 4.4, we have detailed our proposed methods for handling noisy instances in the dataset. In sections 4.5 and 4.6, we have discussed experiment design and results, covering the limitations of our work. Finally, in the last section (Section 4.7), we have presented the conclusion of this work and future research directions.

4.2 Literature Review

4.2.1 Relation Classification from Noisy and Imbalanced Dataset

Given TACRED's noisy and imbalanced nature, a few works have been proposed recently to address these challenges. Song et al. [69] proposed a "classifier-adaptation knowledge distillation framework (CAKD)" for relation extraction and event detection to learn from highly imbalanced datasets. Their work is motivated by the large difference in the number of positive and negative samples in the dataset. The CAKD approach led to a significant performance improvement for all the underlying models. Park et al. [70] proposed a

curriculum learning-based framework based on cross-review method [71]. They have used pre-trained RoBERTA [50], and have achieved the state-of-the-art performance on TACRED.

4.2.2 Dataset Evaluation and Analysis

Despite the use of the leading language models pre-trained on massive corpora, the performance on relation classification is restricted to around 75% F1-Score[1] [6]. This motivated the research community to evaluate TACRED dataset. Alt et al.[6] in their work, trained 49 different RE models, and ranked the test and dev examples following the number of misclassification. They selected 5000 most misclassified examples for further evaluation from linguistic experts. They identified several types of annotation errors and corrected 50% of those examples, which leads to performance improvement of an average of 8% in F1-Score. Model predictions for evaluating datasets have been used earlier, for reading comprehension [72] and sentiment analysis [73] tasks.

In this work, we use models' predictions for TACRED analysis, but in contrast to the earlier studies, *(i)* it does not involve any human expert, *(ii)* it complements model predictions with several additional experiments for in-depth analysis of a dataset.

4.2.3 Dataset Reannotation

While Alt et al. [6] has highlighted the flaws in the dataset, their analysis is subjected to selected samples of test and dev sets only. So, when the supervised datasets have several annotation flaws, an obvious solution is to either detect mislabeled examples or reannotate the entire dataset. ReTACRED [8] did a comprehensive evaluation of TACRED and employed the cost-efficient and improved annotation guidelines for crowd-sourced reannotation of TACRED. After reannotation, 22.1% of labels differed from the original dataset, and the F1-score on the new dataset improved on average by 13%.

The majority of the earlier work, using the model's prediction for detecting noisy instances, focused on eliminating misclassified instances [74]. In contrast to the existing studies, this work focuses on the automated reannotation of the RC dataset TACRED as well. To the best of our knowledge, this is the first work addressing the automated reannotation of the RC dataset TACRED.

[1] https://paperswithcode.com/sota/relation-extraction-on-tacred

Data Split	# Positive	#Negative	# Total
Train	13012	55112	68124
Dev	5436	17195	22631
Test	3325	12184	15509

Table 4.1 Number of instances across different splits of TACRED.

Model	Precision	Recall	F1-score	Pos Acc	Neg Acc	Neg-Pos Accuracy
PALSTM	67.8	64.6	66.2	64.6	92.6	28
LSTM	65.4	61.6	63.4	61.6	92.6	31
BiLSTM	65.2	60.4	62.7	60.4	92.8	32.4
CGCN	71.2	62.6	66.7	62.6	94.2	31.6
GCN	69.2	60.3	64.5	60.3	93.8	33.5

Table 4.2 Baseline Model performance on TACRED. Pos Accuracy and Neg Accuracy are model's accuracy on predicting positive and negative examples respectively.

4.3 TACRED Analysis

TACRED is a sentence-level dataset where each example contains a sentence and a pair of entities. One entity represent the *subject entity* and the other represents the *object entity*. Each example is labelled with one of the 41 relation labels (positive classes) or `no_relation` (negative class). Table 4.1 gives the summary statistics of TACRED across the three splits of training, dev, and test sets. We can observe that the negative class accounts for almost 80% of the entire dataset.

Since its introduction 4 years ago, numerous deep learning models based on LSTM [7], graph convolutional neural network [57, 75], and language models [9, 49, 51] have been proposed and benchmarked on it. Despite the models' increasing complexity, performance on TACRED has been restricted to around 75% F1 score. In recent years, some of the works [6] and [8] have attempted to manually analyze and re-annotate TACRED and have achieved significant improvements in the respective updated datasets.

We first analyze the performance of a few baseline models, including PALSTM [7] and CGCN [57], and observe that model's accuracy in predicting negatively labeled data is significantly higher in comparison to positively labeled datasets (Table 4.2), a common phenomenon on TACRED for other models [9, 51] as well. This makes us question, "Where do models go wrong in accurately predicting positive examples?"

To better understand the model's predictions in an attempt to answer the above question, we examine the prediction results using a confusion matrix. *Confusion matrix* is a 2-d grid, where each row represents the predicted labels' distribution for examples belonging to a

relation class, for example, in Figure 4.3, row associating relation `per:title` shows that 18.2% of examples with ground truth `per:title` in the test set is predicted as `no_relation` by the CGCN model and remaining 81.8% are predicted correctly. In both the confusion matrices in Figures 4.2 and 4.3, we observe that for the considerable number of relations (20 for PALSTM and 21 for CGCN), `no_relation` is predicted for more than 40% of its example (entries in the first column above 40%), which indicates either of the following:

1. The chosen models are weak and could not learn to differentiate between different classes
2. Model's incompetence to learn due to high imbalance in the dataset.
3. There are a significant overlap between positively labeled (one of the 41 relation classes) examples with negative examples. In other words, instances are correctly labeled but the context is very similar.
4. There are substantial relation examples incorrectly annotated as `no_relation`.

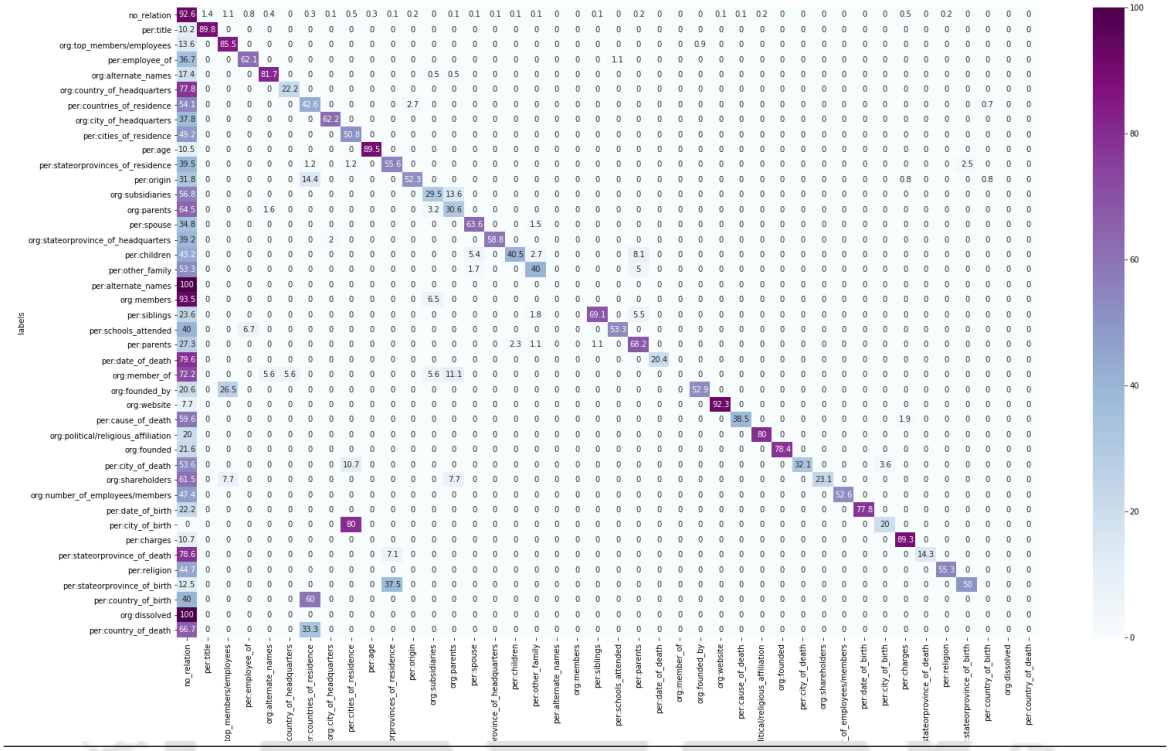
We perform further analyses to establish which among the above hypotheses are likely to be correct or have a significant impact on performance.

4.3.1 Positive Relation Classification Analysis

We trained the baseline models, excluding `no_relation` examples, and generated a confusion matrix to comprehend the models' relation learning potential. We observe that the majority of diagonal elements (correct prediction; 29 and 31 relations for PALSTM and CGCN respectively) in the Figures 4.4 and 4.5 are equal to or above 50%. The overall performance results are shown in Table 4.3. Contrasting these results with the results (Table 4.2 and Figures 4.2 and 4.3) of the models' performance using the entire dataset indicates that the models are capable of learning relations despite fewer training examples and hence confirm the models' competence. There is confusion among a few (positive) relation classes, like `per:city_of_birth` and `per:cities_of_residences`; `org:founded_by` and `org:top_members/employee`, but they are expected as one of the relations is a subset of the other relation.

Based on the above discussion, we can claim that even simpler models such as PALSTM and CGCN are capable of learning relation boundaries if the dataset is relatively clean and has a balanced set of examples for all the classes. Another set of analyses further examines the reason behind the low performance in the following subsections.

Fig. 4.2 Confusion Matrices for PALSTM and model; number indicates the percentage. Rows represent the ground-truth labels and columns represent predicted labels.

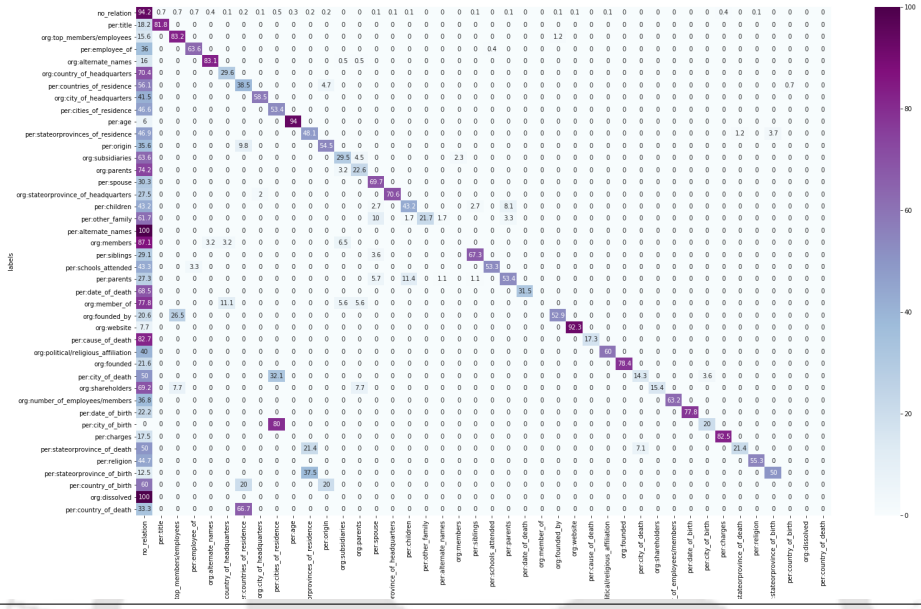


4.3.2 Downsampling

In the previous subsection, we have seen that models are capable of learning relation boundaries among positively labeled instances in the absence of negative examples. This rules out the models' incompetence to learn efficiently. To further assess models' capabilities to learn precisely from imbalanced data, we explore downsampling the number of negative examples. We experiment by randomly removing a portion of the negative samples from the training and development set and not modifying the evaluation dataset. If the sampling ratio is 3:5, that implies that out of every five `no_relation` examples parsed, three are taken into the training/development set. As we move from sampling all the negative examples (ratio of 5:5) to none of the examples (0:5), the accuracy of predicting positive examples (recall) increases dramatically (Figure 4.6) for all the models.

On observing the trend from 0:5 to 5:5 across baselines, the effect of the imbalance in data is quite apparent. However, if we consider a jump of 20% in negative examples (i.e. 1:5 ratio), *(i)* the number of positive and negative examples is almost proportionate (13012

Fig. 4.3 Confusion Matrices for CGCN and model; number indicates the percentage. Rows represent the ground-truth labels and columns represent predicted labels.



Model	F1-Score
PALSTM	88.2
LSTM	87.2
BiLSTM	86.6
CGCN	88.7
GCN	87.2

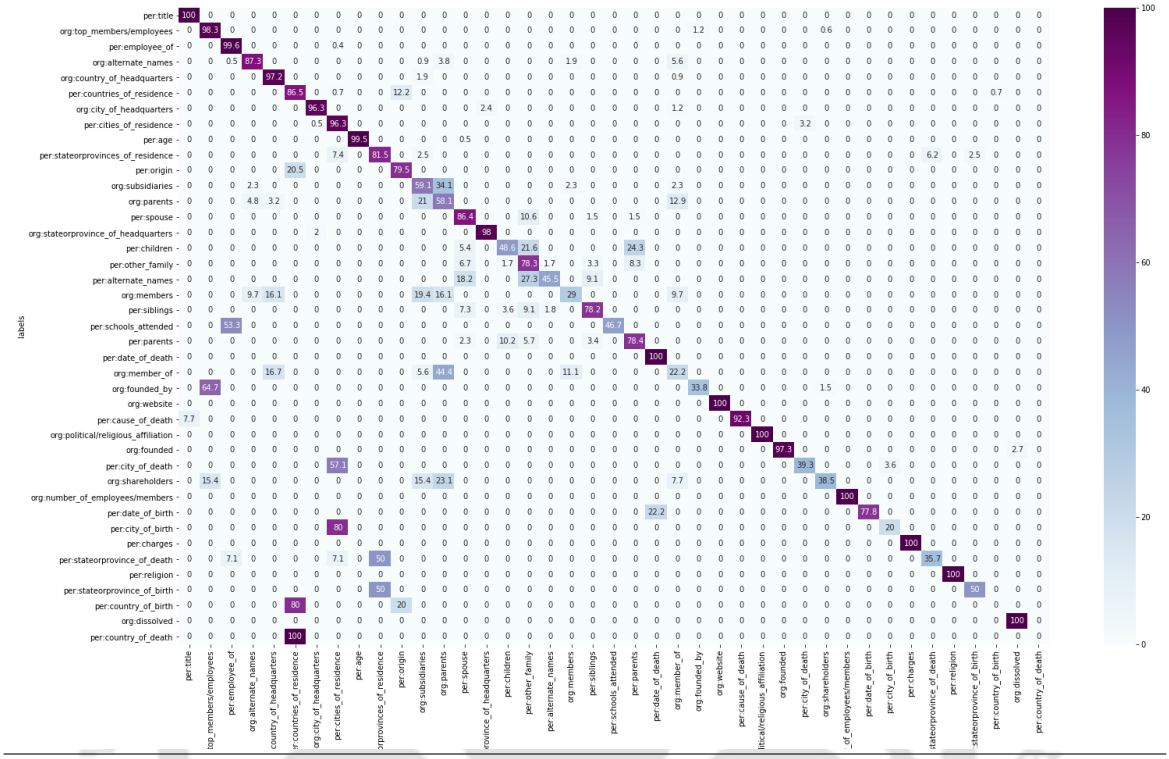
Table 4.3 Performance of baseline models on TACRED excluding `no_relation` examples. Each number is reported as a percentage.

positive examples and 11022 negative examples) and *(ii)* the recall falls close to 10% across all the baseline models. This sharp drop in the model's potential to learn from positive examples indicates that introducing negative examples brings in extra challenges apart from a disproportionate number of samples.

4.3.3 Binary Classification

To further confirm the observations from downsampling, we formulated the RC on TACRED as a binary classification problem. All the positive relations examples are grouped under a single label: `relation`. Table 4.4 shows the performance of all baseline models for binary classification. We assume that identifying whether a relation exists or not is a simpler

Fig. 4.4 Confusion Matrices for PALSTM models excluding no_relation; number indicates the percentage. Rows represent the ground-truth labels and columns represent predicted labels.



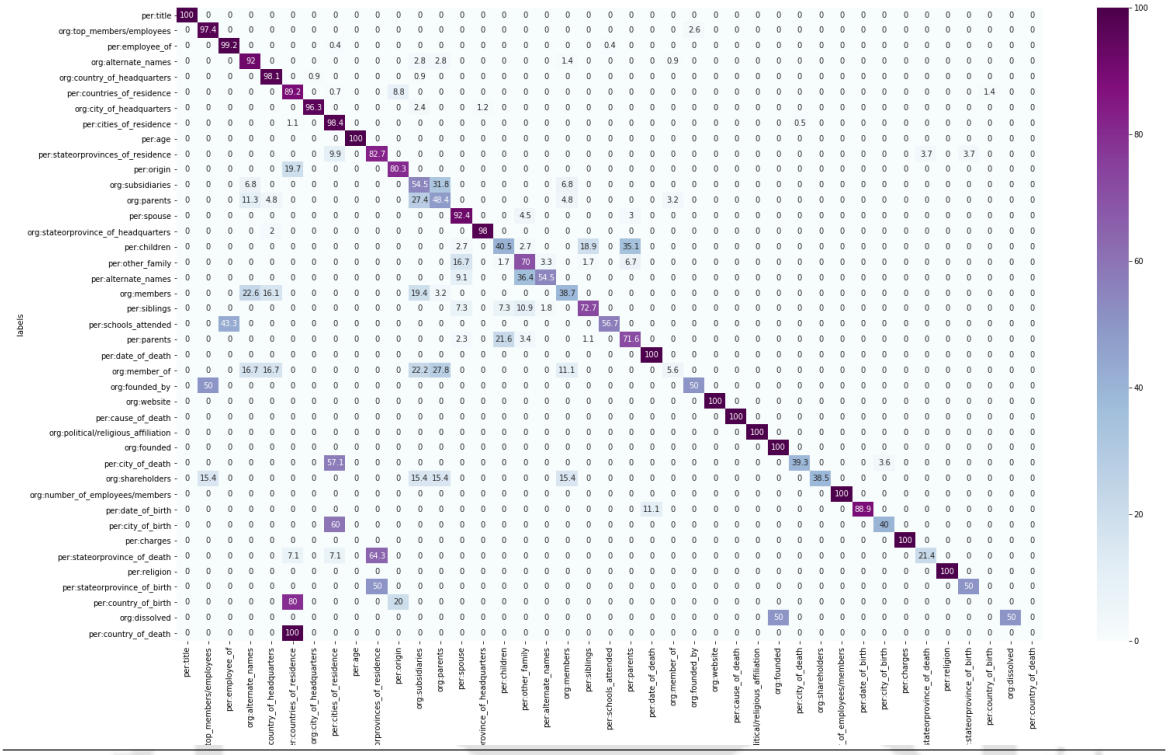
classification problem than identifying the exact relation class from 41 relation labels or no relation.

On average, recall improves by 5.4 percent and F1-score improves by 5.2 percent across different baseline models (refer to respective columns in Table 4.2 and Table 4.4). This meagre change in performance hints that *(i)* class imbalance may not be the prominent reason, and *(ii)* there are certain negative examples that either share a similar context with positive examples or there are a significant number of incorrectly labeled negative examples.

4.3.4 t-SNE Plots

Above discussed analyses indicate that the presence of a disproportionate number of examples labeled with *no_relation* in the dataset may not be the prominent reason for the low performance of models. The low performance can be attributed to the nature of negative examples in the dataset. Therefore, to carefully study the characteristics of those examples, we visualize their representation using *t-SNE visualization*. t-distributed stochastic neighbour

Fig. 4.5 Confusion Matrices for CGCN models excluding no_relation; number indicates the percentage. Rows represent the ground-truth labels and columns represent predicted labels.



embedding (t-SNE) [76] is a non-linear dimensionality reduction algorithm for visualising high dimensional data points by mapping them to a lower dimension, preserving the information between the points. We use t-SNE plots to project embeddings learned for each relation from a random batch of 3000 test set examples to a 2-d plane. From Figures 4.7a and 4.7b, we observe that points (black dots) associating with `no_relation` are all over the space which is not the case in Figures 4.8a and 4.8b, where negative examples are excluded. This shows that embeddings or representations generated by models for negative examples are quite similar to the embeddings of positive ones. This analysis indicates the likelihood of

Model	Precision	Recall	F1-score
PALSTM	71.6	68.5	70
LSTM	72.5	65.8	69
BiLSTM	73.4	66.1	69.6
CGCN	73.8	68	70.8
GCN	71.4	67.1	69.2

Table 4.4 Performance of baseline models as Binary Classification (relation or no_relation) on TACRED. Each number is reported as a percentage.

Fig. 4.6 Effect of downsampling `no_relation` from dataset on positive accuracy (Recall). 1:5 implies, 1 out of 5, i.e. only 20% of negative examples are selected.

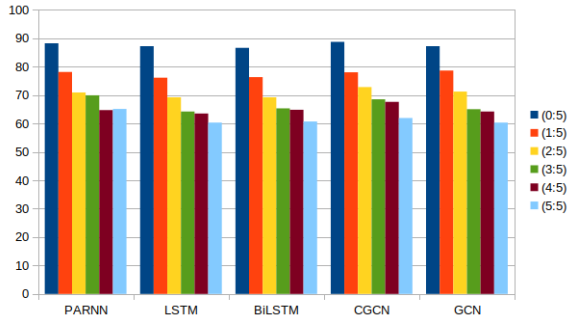
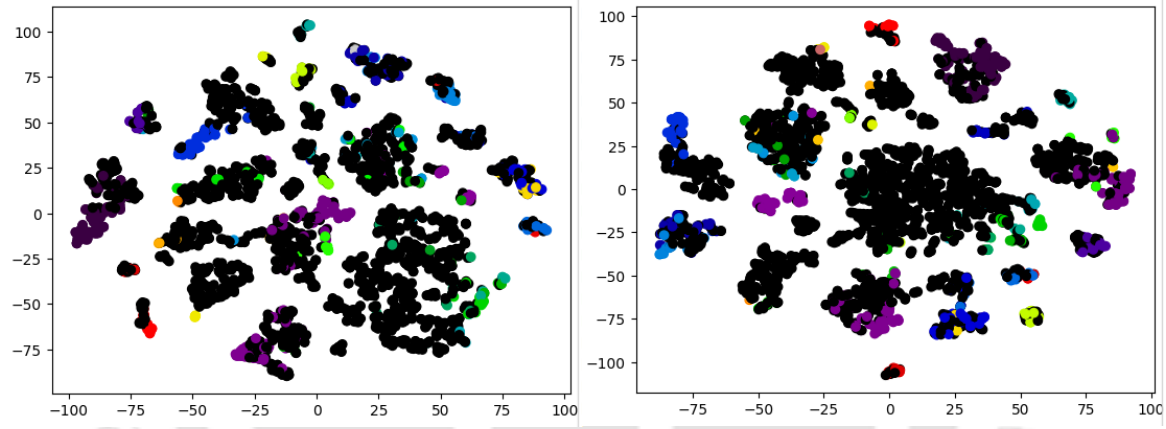


Fig. 4.7 t-SNE plots for a batch of 3000 test examples. Black dots represents instances of *no_relation*. Different colour represents different relation label.



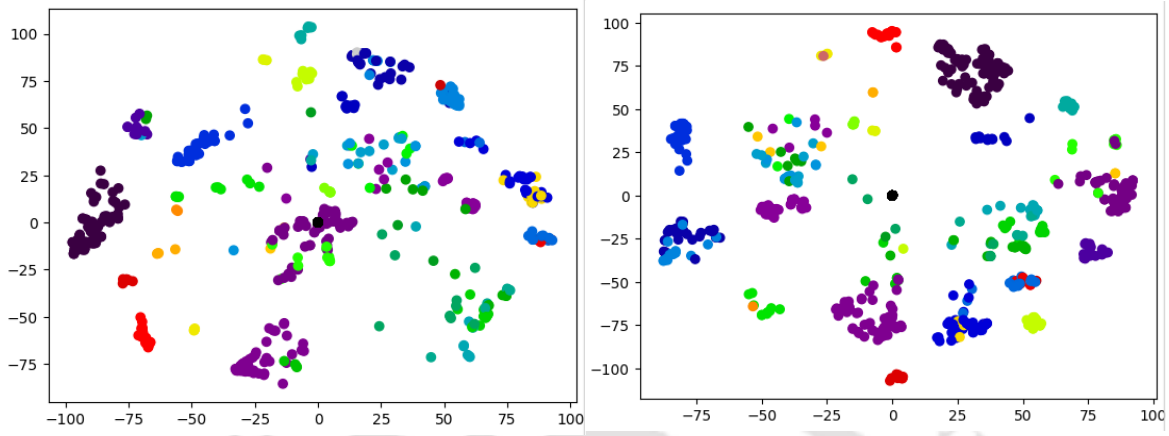
(a) CGCN with no_relation (b) PALSTM with no_relation

either one or more of the following cases: *(i)* negative examples share context with examples from several positive classes, *(ii)* negative examples do not have any specific bias towards a particular positive class and *(iii)* some negative examples are potentially noisy in the sense that they are incorrectly labeled as `no_relation`.

4.3.5 Top-k Evaluation

In the previous section, we analyzed inconsistencies in TACRED annotation. We have shown that models are inefficient in learning from positive samples due to the noisy nature of negative examples. In this section, we further examine the same from the model's predictions by looking at the confidence score assigned to each relation for a test example. We observed that the difference between the first and second predictions is often minimal, with the second-

Fig. 4.8 t-SNE plots for a batch of 3000 test examples. Different colour represents different relation label.



(a) CGCN without no_relation (b) PALSTM without no_relation

highest predicted label frequently being the actual ground truth label. This observation motivated us to inspect the results further, and in most cases, the top two labels contain a `no_relation` label. We hypothesized that even when the model correctly predicts a label, it gets confused by the `no_relation` label. This observation corroborates the findings from the confusion matrix.

To further empirically validate our hypothesis, we modified the evaluation criteria. If the ground truth were present in the model's $top-K$ predictions for a particular example, it would be counted as a correct prediction. In case the ground truth was not present in the $top-K$ predictions, the scorer would count it as a wrong prediction and carry out other calculations with the highest probability prediction. We re-ran experiments, taking $K = 2, 3$. The results of the same have been outlined in Table 4.5 for $K = 2$ and in Table 4.6 $K = 3$. We have also depicted the percentages of examples where the first, second, and third predictions are correct and the percentage of examples where the ground truth is not present in any of the guesses under the columns *Top1*, *Top2*, *Top3*, and *Wrong* respectively.

Based on all the above analyses (Section 4.3.1-4.3.5), we conclude that

1. Even the simplest model such as LSTM are capable of learning relation boundaries (Section 4.3.1).
2. Apart from disproportionate examples in the dataset, context overlapping and incorrect labeling of the sample as `no_relation` are the primary concern (Section 4.3.2-4.3.5, particularly Section 4.3.4).
3. Models' performance suffer primarily due to noise originating from the negative examples in the dataset (Section 4.3.3, 4.3.5).

Model	Precision	Recall	F1-score	Top 1	Top 2	Wrong
PALSTM	95.6	89	92.2	86.6	10.7	2.7
LSTM	93.3	88.3	90.8	86	10.9	3.1
BiLSTM	92.6	87.1	89.8	85.9	10.7	3.4
CGCN	96.1	89.8	92.9	87.4	10.2	2.4
GCN	95.7	88.3	91.8	86.6	10.6	2.8

Table 4.5 Performance of baseline modes as top k(k = 2) classifier on TACRED with percentage of top1, top2, and wrong guesses. Each number is reported as a percentage.

Model	Precision	Recall	F1-score	Top1	Top 2	Top3	Wrong
PALSTM	98.4	94.9	96.6	86.6	10.7	1.5	1.2
LSTM	96.9	94.2	95.6	86	10.9	1.6	1.4
BiLSTM	96.9	93.7	95.3	85.9	10.7	1.8	1.6
CGCN	99	95.3	97.1	87.4	10.2	1.4	1
GCN	98.9	94.9	96.9	86.6	10.6	1.6	1.2

Table 4.6 Performance of baseline modes as top k(k = 3) classifier on TACRED with percentage of top1, top2, top3, and wrong guesses. Each number is reported as a percentage.

4.4 Methodology: Handling Noisy Instances

Analyses in the previous section indicate that noisy instances are primarily present in the no_relation or negative relation class. While some of the noisy instances are likely to have a similar context as one or more instances from one of the positive relation classes, the others are incorrectly labeled. This section proposes two different methodologies for identifying noisy instances. For both the methodologies, we explore elimination and reannotation with appropriate labels to evaluate their impact on model performance. We discuss the two strategies below.

4.4.1 Intrinsic Strategy (IS)

We refer to the first strategy as *intrinsic strategy (IS)*, as it utilizes only the given data and model. This strategy does not depend on the external data or the results from other models.

The intrinsic strategy assumes that *substantial majority of positive samples, i.e., samples belonging to any one of the positive relation classes, are clean.* The assumption is based on our observations that *(i)* a significant number of predictions for instances from the majority

of positive relation classes is `no_relation`, and *(ii)* models are capable of learning relation boundary for a majority of the positive relations if trained without no_relation class. With this assumption, when a model M makes a false-negative prediction, i.e., predicts `no_relation` for a positive labeled sample p_i with label l_i, it implies that there are considerable number of negative training samples $ntr_1, ..., ntr_k \in N_{tr}$ that either share similar context or are incorrectly labeled with `no_relation`.

Once the model is trained, we pass the target data for evaluation and collect all the instances with false-negative predictions. We refer to this collection as seed set S. We pass this set S to the Algorithm 2, which uses cosine similarity for finding the k-nearest-neighbour from negative set N_{tr}. Once we have k most similar negative examples $ntr_1, ..., ntr_k \in N_{tr}$ for each seed examples $s_i \in S$, we add them to elimination list *eliminate_sids* if they have been not already included. Similarly, we add those examples as keys to the relabel dictionary *relabel_sids* with ground_truth label l_i of s_i as value (relation to be reannotated with). In case, if ntr_j is already present in *relabel_sids*, we assign the label with the highest similarity score.

4.4.2 Extrinsic Strategy (ES)

Our first strategy of using TACRED to identify the noisy examples was based on a hypothesis that the majority of positive examples are clean. In the second strategy, we took the clean set of sentences to obtain a list of noisy examples from the TACRED. A clean subset of TACRED can be obtained by appointing some experts at a small cost. However, we took the clean set from the ReTACRED [8] test data. As this strategy depends on the external data, we refer to this strategy as an extrinsic strategy (ES).

4.4.2.1 Clean Subset of TACRED

For reannotating TACRED, Stoica et al. [8] have reformulated some of the relations' definitions. Therefore, to avoid any ambiguities, we consider only those relations that can be directly mapped with the original TACRED dataset. The clean set contains 7770 negative examples and 2707 positive examples spanning 24 relation labels. Refer to the original paper for the list of relations with updated guidelines.

Algorithm 2 Algorithm for finding noisy examples for elimination and reannotation using IS

Input: Seed Set S
Parameter: k, N_{tr}, M
Output: $relabel\_sids$, $eliminate\_sids$

Let $t = 0$.
Let $eliminate\_sids = list()$.
Let $relabel\_sids = dict()$.
while $t < len(S)$ **do**
  $rel = ground\_truth(S_t)$
  $pred = M(S_t)$
  $sids = k\_similar\_negative\_examples(S_t, N_{tr}, k)$.
  Let $v = 0$.
  while $v < k$ **do**
    if $sids[v] \notin eliminate\_sids$ **then**
      $eliminate\_sids.append(sids[v])$.
      $relabel\_sids[sids[v]] = rel$.
    else
      if $similarity\_score(sids[v], S_t) > similarity\_score(sids[v], S_{current\_high})$ **then**
        $relabel\_sids[sids[v]] = rel$.
      end if
    end if
    $v = v + 1$.
  end while
  $t = t + 1$.
end while
return $relabel\_sids, eliminate\_sids$

We use separate algorithms for finding noisy examples for elimination and reannotation. In both cases, we use the representation obtained from the model trained on the TACRED training set. They are described next.

4.4.2.2 Finding noisy examples for elimination

We use Algorithm 3 for finding noisy examples for elimination. This algorithm takes the entire TACRED dataset (training, dev, and test sets) T as an input. It iterates over each example $t_i \in T$ and finds k-nearest-neighbour from clean set C using cosine similarity. Assuming the example t_i has label l_i, the algorithm checks if none of those k examples $c_1, c_2, ..., c_k \in C$ has the same label as l_i, it adds the example t_i to the list of noisy examples for elimination $eliminate\_sids$. In other words, if label l_i given in TACRED does not match

Algorithm 3 Algorithm for finding noisy examples using ES for elimination.

Input: TACRED Examples T
Parameter: k, C
Output: $eliminate\_sids$

```
Let t = 0.
Let eliminate_sids = list().
while t < len(T) do
    rel = ground_truth(T_t)
    sids = k_similar_clean_examples(T_t, C, k).
    Let v = 0.
    Let count = 0.
    while v < k do
        if ground_truth(sids[v]) == rel then
            count = count + 1.
        end if
        v = v + 1.
    end while
    if count == 0 then
        eliminate_sids.append(T_t).
    end if
    t = t + 1.
end while
return eliminate_sids
```

with even a single k nearest-neighbors from the clean set, then it is likely that l_i may not be the correct label.

4.4.2.3 Finding noisy examples for reannotation

We use Algorithm 4 for finding noisy examples for reannotation. This algorithm takes TACRED dataset (training, dev and test sets) T as an input. It iterates over each example $t_i \in T$ and find k-nearest neighbour from clean set C using cosine similarity. Assume the example t_i has label l_i and its k-nearest-neighbours from the clean set C are $c_1, c_2, ..., c_k$. The algorithm considers t_i as a noisy instance if the following conditions are met: $\forall j \in \{1, \ldots, k\}$, label of $c_j = l$, and $l \neq l_i$. In other words, an instance t_i is potentially noisy if all of its nearest neighbours belong to the same class but does not match with the class of t_i. We add t_i to the dictionary of noisy examples for reannotation $relabel\_sids$, where key is sentence id of t_i and value is l, the label of nearest-neighbours.

Algorithm 4 Algorithm for finding noisy examples using ES for reannotation.

Input: TACRED Examples T
Parameter: k, C
Output: *relabel_sids*

Let $t = 0$.
Let $relabel\_sids = list()$.
while $t < len(T)$ **do**
  $rel = ground\_truth(T_t)$
  $sids = k\_similar\_clean\_examples(T_t, C, k)$.
  Let $v = 0$.
  Let $rel\_lst = list()$.
  while $v < k$ **do**
    if $ground\_truth(sids[v]) \neq rel$ **then**
      $rel\_lst.append(ground\_truth(sids[v]))$.
    end if
    $v = v + 1$.
  end while
  if $check\_all\_relations\_are\_same(rel\_lst)$ **then**
    $relabel\_sids[T_t] = rel\_lst[0]$.
  end if
  $t = t + 1$.
end while
return *relabel_sids*

4.5 Experiments

Following the first approach discussed in subsection 4.4.1, we conduct two sets of experiments, *(i)* we find noisy training examples only and eliminate/reannotate them, *(ii)* we find noisy examples from all three sets (train, dev, test) for elimination/reannotation. In the second set of experiments, the model is always trained using training data only.

4.5.1 Baseline Models and Hyper-parameters

All our analysis are based on the following baseline models: PALSTM [7], CGCN [57], LSTM, BiLSTM, and GCN. All the input word vectors are initialized using pre-trained GloVe vectors [12]. We consider 300-dimensional vectors for CGCN and GCN and 200-dimensional vectors for LSTM, BiLSTM, and PALSTM. For other tags such as POS and NER, 30-dimensional vector representations are considered.

For training GCN and CGCN, we use the same set of hyper-parameters as in [57]. We use 200 nodes in LSTM and feedforward hidden layers. We use 2 GCN layers and 2 feedforward layers, SGD as an optimizer, with an initial learning rate of 1.0 which is reduced by a factor of 0.9 after epoch 5. We train the model for 100 epochs. We use word dropout probability 0.04 and dropout probability of 0.5 for LSTM layers.

For training PALSTM, LSTM, and BiLSTM, we follow [7] for hyper-parameters. We have used 2 layered stacked LSTM layers for all the models with the hidden size of 200. We use *AdaGrad* with a learning rate of 1.0 which is reduced by a factor of 0.9 after the 20th epoch. We have trained the model for 30 epochs. We use word dropout probability 0.04 and dropout probability of 0.5 for LSTM layers.

4.5.2 Evaluation Models

Apart from the two SOTA models (PALSTM and CGCN) among baselines, we use two recent SOTA RE models based on pre-trained large-scale language models for evaluation of elimination and reannotation using ES.

As recent SOTA RE models, we use SpanBERT [9], which employs a bi-directional language model similar to BERT [24] pre-trained on span-level, and a recent model [50] that uses pre-trained BERT along with typed entity marker to better highlight the entity information for RE. For both the methods we use *base-cased* version along with the set of hyper-parameters as used in the respective works [9, 50].

4.5.3 TACRED Variant for Evaluation

Other than TACRED, we have also shown evaluation results on TACRev [6] eval data. [6] trained 49 different RE models to identify the most challenging examples from dev and test sets. They identified 1923 examples as challenging. 960 from that set were later reannotated by linguistic experts. This cleaner version of TACRED is referred to as TACRev by [6].

4.6 Results and Discussion

In the following subsections, we discuss performance of models under different experiment settings.

4.6.1 Performance Evaluation

Intrinsic Strategy (IS) on TACRED test set: Results for all considered models, trained after both elimination and reannotation, on the TACRED test set are presented in row 2 of each model block in Table 4.7 and Table 4.8 respectively. We observe that recall gets boosted significantly but at the cost of precision. This reflects that both elimination and reannotation make the model better at predicting examples from positive relation classes.

IS on the cleaned TACRED test set: In the above analysis, evaluation was done on the original TACRED evaluation set. Since studies have indicated that this set is also noisy. Any evaluation of such a noisy set may not give us a true picture. To evaluate both our elimination and reannotation strategies on cleaner evaluation datasets, we performed another experiment by passing dev and test sets to the same algorithm (Algorithm 2). Instead of negative training samples, negative dev and test samples were used to identify noisy examples from the two sets respectively. As discussed earlier, the model is trained using training data only. Dev and test set samples are fed to the model to obtain corresponding representations.

Performance of SOTA models trained after elimination and reannotation on cleaned TACRED, TACRED-E, and TACRED-R for elimination and reannotation respectively are presented in Tables 4.7 and 4.8 respectively. Third row of each model block contain results on original TACRED test set and fourth row contain results on new TACRED-E (Table 4.7) or TACRED-R (Table 4.8) test set. The performance of the model went down significantly after training on reannotated TACRED. However, the model performance shows significant improvement in F1-score after training on eliminated TACRED. Based on the result, we hypothesize that our approach of finding noisy examples using nearest neighbors of false-negative model prediction can identify noisy examples. But, reannotated labels may not be correct and hence, the model is unable to take advantage of them.

Extrinsic Strategy(ES) on TACRED test set: In our second strategy, we use subset of ReTACRED [8] test set as clean dataset for identifying noisy instances using different algorithms for elimination (Algorithm 3) and reannotation (Algorithm 4). Performance of all the models trained on eliminated and reannotated negative examples of TACRED are presented in Tables 4.9 and 4.10 respectively. The second row of each model block contains results on the original TACRED test set and the third row contain results on new TACRED-EN and TACRED-RN test sets. On both the datasets, models have shown improved performance in recall and F1-score, despite a very small fraction of updates in the TACRED.

Model	# Elimination	Evaluation Data	Precision	Recall	F1-Score
PALSTM	baseline	TACRED	67.8	64.6	66.2
	5334, 0, 0	TACRED	63.6	65.7	64.7
	5334, 3092, 0	TACRED	62.8	66.2	64.4
	5334, 3092, 2038	TACRED-E	71.9	66.2	68.9
CGCN	baseline	TACRED	71.2	62.6	66.7
	5321, 0, 0	TACRED	66.6	67.4	67
	5321, 3133, 0	TACRED	63	70.8	66.7
	5321, 3133, 2043	TACRED-E	73.1	70.8	71.9

Table 4.7 Performance of SOTA models after elimination of noisy negative examples identified using Algorithm 2 following IS on TACRED evaluation data. TACRED-E represents TACRED after the elimination of noisy examples. Except for baseline results, all the models are trained on the TACRED-E train set. **# Elimination** represents the number of examples eliminated from (train, dev, test).

Model	# Reannotation	Evaluation Data	Precision	Recall	F1-Score
PALSTM	baseline	TACRED	67.8	64.6	66.2
	5334, 0, 0	TACRED	56.7	68.8	62.2
	5334, 3092, 0	TACRED	52.8	73	61.3
	5334, 3092, 2038	TACRED-R	66.7	57.2	61.6
CGCN	baseline	TACRED	71.2	62.6	66.7
	5321, 0, 0	TACRED	59.4	71.2	64.8
	5321, 3133, 0	TACRED	55.4	73.4	63.1
	5321, 3133, 2043	TACRED-R	68.8	56.5	62

Table 4.8 Performance of SOTA models after reannotation of noisy negative examples identified using Algorithm 2 following IS on TACRED evaluation data. TACRED-R represents TACRED after the reannotation of noisy examples. Except for baseline results, all the models are trained on the TACRED-R train set. **# Reannotation** represents the number of examples reannotated from (train, dev, test).

Model	Train Data	# Elimination	Evaluation Data	Precision	Recall	F1-Score
PALSTM	TACRED	baseline	TACRED	67.8	64.6	66.2
	TACRED-EN	1568, 852, 0	TACRED	67.1	66.8	67
	TACRED-EN	1568, 852, 481	TACRED-EN	73.4	66.8	69.9
	TACRED-ENP	2669, 1543, 0	TACRED	68	63.9	65.9
	TACRED-ENP	2669, 1543, 741	TACRED-ENP	74.9	68.3	71.5
CGCN	TACRED	baseline	TACRED	71.2	62.6	66.7
	TACRED-EN	1878, 961, 0	TACRED	67.8	67	67.4
	TACRED-EN	1878, 961, 466	TACRED-EN	74.4	67	70.5
	TACRED-ENP	2997, 1610, 0	TACRED	68.2	66.8	67.5
	TACRED-ENP	2997, 1610, 721	TACRED-ENP	75.2	70.7	72.9

Table 4.9 Performance of SOTA models trained after eliminating noisy examples following Algorithm 3. TACRED-EN represents TACRED after the elimination of noisy negative examples. TACRED-ENP represents TACRED after the elimination of noisy negative and positive examples. **# Elimination** represents the number of examples eliminated from (train, dev, test).

We further evaluated this strategy by finding noisy positive examples as well using the same algorithms. After eliminating and reannotating such examples from TACRED, the models performance improved further (Tables 4.9 and 4.10). The fourth row contains results on the original TACRED test set and the fifth row contain results on the new TACRED-ENP and TACRED-RNP test sets.

ES on cleaned TACRED test set: As discussed earlier, cleaning only the training set will not give us a true picture if the evaluation data is still noisy [77]. Thus, we have used cleaned evaluation data for reporting model performances. Here, we have also included two models based on pre-trained large-scale language models. Table 4.11 presents the performance of the four SOTA models for TACRED and TACRev. Our proposed strategy ES for eliminating/reannotating noisy examples using the clean subset of the dataset has shown improvement in F1-score across all the variants for all 4 models. We observe an average performance improvement of around 4% for elimination and reannotation of only negatively labeled data on TACRED and 2.6% and 1% for elimination and reannotation of only negatively labeled TACRev data. While eliminating and reannotaing both positive and negative labeled noisy examples, we have an average improvement of 7.2% and 6% for elimination and reannotation respectively on TACRED and 4.2% and 1.7% for elimination and reannotation respectively on TACRev. All the models got significant performance improvement, sometimes even more than 6%, across all variants of the modified evaluation data following our proposed strategies. BERT model remained the best performing model even on modified evaluation data.

Model	Train Data	# Reannotation	Evaluation Data	Precision	Recall	F1-Score
PALSTM	TACRED	baseline	TACRED	67.8	64.6	66.2
	TACRED-RN	1354, 768, 0	TACRED	65.8	66.8	66.3
	TACRED-RN	1354, 768, 409	TACRED-RN	74.1	67	70.3
	TACRED-RNP	2284, 1351, 0	TACRED	67.7	64.9	66.3
	TACRED-RNP	2284, 1351, 634	TACRED-RNP	75.6	68.3	71.8
CGCN	TACRED	baseline	TACRED	71.2	62.6	66.7
	TACRED-RN	1642, 843, 0	TACRED	67.9	66.9	67.4
	TACRED-RN	1642, 843, 389	TACRED-RN	76	67.1	71.3
	TACRED-RNP	2540, 1323, 0	TACRED	69.2	65.2	67.1
	TACRED-RNP	2540, 1323, 573	TACRED-RNP	77.1	68	72.3

Table 4.10 Performance of SOTA models trained after reannotating noisy examples following Algorithm 4. TACRED-RN represents TACRED after the reannotation of noisy negative examples. TACRED-RNP represents TACRED after the reannotation of noisy negative and positive examples. **# Reannotation** represents the number of examples reannotated from (train, dev, test).

		PALSTM			CGCN			SpanBERT			BERT		
Datset	Variant	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score
TACRED	Original	67.8	64.6	66.2	71.2	62.6	66.7	66.4	66.1	66.3	71.1	71.4	71.2
	Eliminate-Neg	73.4	66.8	69.9 (+3.7)	74.4	67	70.5 (+3.8)	77.2	65.5	70.9 (+4.6)	79.6	71.3	75.2 (+4)
	Eliminate-Neg&Pos	74.9	68.3	71.5 (+5.3)	75.2	70.7	72.9 (+6.2)	80.5	72.5	76.3 (+10)	84.5	73	78.3 (+7.1)
	Reannotate-Neg	74.1	67	70.3 (+4.1)	76	67.1	71.3 (+4.6)	76.8	64.5	70.1 (+3.8)	79	71.7	75.2 (+4)
	Reannotate-Neg&Pos	75.6	68.3	71.8 (+5.6)	77.1	68	72.3 (+5.6)	76.5	68.8	72.5 (+6.2)	78.5	76.9	77.7 (+6.5)
TACRev	Original	73.7	74.8	74.3	77.5	72.6	75	71.3	75.6	73.4	75.9	81.1	78.4
	Eliminate-Neg	75.4	76.9	76.1 (+1.8)	75.9	76.9	76.4 (+1.4)	79.3	75.7	77.5 (+4.1)	81.1	81.7	81.4 (+3)
	Eliminate-Neg&Pos	76.7	75.3	76 (+1.7)	77	78.7	77.9 (+2.9)	81.9	78.9	80.4 (+7)	86.3	80.7	83.4 (+5)
	Reannotate-Neg	74.7	74.6	74.7 (+0.4)	76.9	75.2	76 (+1)	77.7	71.5	74.5 (+1.1)	79.8	80	79.9 (+1.5)
	Reannotate-Neg&Pos	76.8	73.9	75.4 (+1.1)	78.4	74.5	76.4 (+1.4)	77.2	73.2	75.2 (+1.8)	79.2	82.5	80.8 (+2.4)

Table 4.11 Evaluation of our proposed dataset elimination/reannotation strategy based on 4 SOTA models.

Model	Classifier	Precision	Recall	F1-Score	Positive Accuracy	Negative Accuracy	Accuracy
PALSTM	Linear	66.5	65.7	66.1	65.7	92.2	86.5
	Bayesian Linear	68.3	62.9	65.5	62.9	93.2	86.7
CGCN	Linear	71.6	60.8	65.8	60.8	94.5	87.3
	Bayesian Linear	70.6	63.5	66.8	63.5	93.8	87.3

Table 4.12 Baseline models' performance comparison. For each model, "Linear" and "Bayesian Linear" indicate the final layers as Linear and Bayesian Classifier respectively.

Qualitative Comparison of IS and ES: Our analysis in section 4.3 indicates that negative examples are the main source of noise in the dataset. So, we assumed that majority of the positive instances are clean for the *IS*, whereas no such assumptions were made for the *ES*. *IS* relies only on the model's prediction, whereas *ES* involves external help in obtaining the clean subset. As a result, *IS* guarantees cost-efficiency whereas *ES* ensures higher accuracy. Furthermore, using *IS*, the same algorithm can be used for finding instances for elimination and reannotation. Whereas for *ES*, we have separate algorithms for finding instances for elimination and reannotation.

4.6.2 Analysis of Noisy Sample

Uncertainty Analysis: To account for the uncertainty of deep learning models, we experimented by replacing a *linear* classifier layer with a *Bayesian* classifier layer. The results from using the two baseline models are shown in Table 4.12. As it can be seen, the two models gave different results. The performance of the PALSTM model deteriorated, while the CGCN model improved its performance in terms of F1-score by almost 1%. One plausible explanation could be that the inconsistency in the results of the two models is due to *aleatoric* uncertainty, i.e., uncertainty in the data [78]. Data uncertainty is considered irreducible, as it corresponds to the inherent property of the dataset [78]. Of course, this warrants further detailed analysis, which is beyond the scope of this work.

Furthermore, we also performed a simple analysis on the probability score assigned to the predicted relation label by the Bayesian classifier. The reported result is for the CGCN-Bayesian model (CGCN with Bayesian Linear final layer). However, similar results were observed for the PALSTM-Bayesian model (PALSTM with Bayesian Linear final layer). The probability score across all test instances ranges from 0.2 to 0.99. We consider examples with a probability score for the predicted class of less than 0.8 to be the ones with

higher uncertainty. Such examples can be considered "noisy instances". We compared those instances with noisy instances selected either for elimination or reannotation by the two proposed strategies. We found more than 40% overlap of the noisy instances selected by either of the strategies with the uncertain examples detected by the CGCN-Bayesian model. This analysis indicates that future work can consider combining the uncertainty measures and heuristics to obtain noisy instances of data.

Manual Analysis: To evaluate the quality of examples shortlisted as noisy for elimination or reannotation following the extrinsic strategy, we randomly sampled a hundred examples that are common in identified potentially noisy examples by the PALSTM and CGCN models. We analysed those sentences based on the following error categories: (i) *Falsely labeled as no_relation*, (ii) *Incorrect Span:* Entities span is not properly marked/identified, and (iii) *Wrong Entity Type:* Entity type for one of the argument entities is incorrect.

Out of 100 sampled examples for reannotation, 68 examples labeled with no_relation are incorrectly labeled and they should be labeled with one of the 41 relation labels. 17 examples are falsely identified as noisy. And the remaining can be attributed to the other two error categories: incorrect span and wrong entity type.

Out of 100 sampled for elimination, 62 examples labeled with no_relation are incorrectly labeled. 15 examples are falsely identified as noisy and the remaining can be attributed to error categories such as incorrect span and wrong entity types.

Despite falsely identifying a small percentage as noisy, our approach is capable of providing substantial noisy instances (68 out of 100). Thus, it can be easily scaled for any large dataset.

4.6.3 Robustness of our Approach

All the results discussed in the previous section are based on a single run of a model. Thus, to verify whether those results are not biased by the model parameters, we performed a small experiment. We trained PALSTM another four times with distinct initialization. For each distinct run, we generated corresponding sets of noisy instances for elimination and reannotation. In the comparison of these sets, we observed, *(i)* the number of instances generated for elimination and reannotation is in the range of 453 to 481 and 373 to 409 respectively. *(ii)* more than 50% of instances are common across all the sets for both reannotation and elimination, and *(ii)* more than 70% instances are identified as noisy by more than 3 different models.

The above discussion indicates that the proposed strategies are robust to multiple runs of a single model. Further, modifying the strategies to consider potential noisy instances based on multiple runs is likely to improve the model performance.

4.6.4 Impact on different Relation Labels

The statistical impact of eliminating and reannotating noisy instances using clean data on the relation label set is presented in Table 4.13. All the reported numbers are the intersection of two baseline models CGCN [57] and PALSTM [7]. 11 relations have shown performance improvement and for 4 relation labels performance did not change across all the strategies. There is no common relation with declining performance across all 4 strategies.

4.6.4.1 Impact of eliminating noisy instances:

In our dataset analysis, we have shown that negative instances are the main source of noise in the dataset, and eliminating them has shown performance improvement across both our strategies (Tables 4.7, 4.9). On evaluating the impact of removing negative noisy instances following our best strategy (ES), i.e. using a clean set, 20 relation labels shows improved performance. Moreover, by eliminating noisy positive instances as well, the number of relation labels rises to 23. However, 17 relation labels are common to both the elimination strategies.

4.6.4.2 Impact of reannotating noisy instances:

Although reannotating noisy instances have not shown performance improvement across both our strategies (Tables 4.8, 4.10), using a clean dataset, the performance improvement is significant. On evaluating the impact of reannotating negative noisy instances following our best strategy, i.e. using a clean set, 18 relation labels show improved performance. Moreover, by eliminating noisy positive instances as well, the number of relation labels rises to 19. However, 15 relation labels are common to both the reannotation strategies.

Out of 24 relation labels considered in a clean set other than no_relation, on reannotating only negative instances, performance for 14 relations improved for both the models and for 7 relations, it improved for at least one model. While reannotating both positive and negative instances, for 17 relations performance of both the models improved, and for 4

Strategy	#Rel Performance Improved	#Rel Performance Declined	#Rel Performance Remained Same
Eliminating Negative	20	1	7
Reannotating Negative	18	4	5
Eliminating Neagative & Positive	23	3	6
Reannotating Negative & Positive	19	5	6

Table 4.13 Impact of different strategies on performace of different relation labels. Each number represents intersection of PALSTM and CGCN models.

relations performance improved for at least one model. The above observations establish that including unambiguous instances for relations not covered in the clean set can further improve the model performance.

4.6.5 Limitations

In the extrinsic strategy, we have used the ReTACRED test set as clean data to avoid any form of human intervention. However, to adopt our proposed method, one needs to employ expert annotators to obtain a high-quality clean subset of data. The clean subset can also be obtained by considering high-confidence examples from multiple baseline models. Nevertheless, the quality of the data can be questionable, thus it is left for investigation in future work.

Further, the proposed strategies can overclean the dataset as we have earlier shown that 17 out of 100 examples are falsely identified as noisy. However, we can control this to some extent by taking only the most confident noisy instances across multiple runs of a model. The use of quantitative uncertainty models can be an alternative way to control this as the uncertainty analysis in the section 4.6.2 indicates.

4.7 Chapter Summary

This work presented a model-based characterization of the noise present in the TACRED dataset and two strategies to handle potentially noisy instances. To the best of our knowledge, this is the first work that uses models' prediction and performance to characterize the noisy nature of the RC dataset. Moreover, this work can be easily automated and generalized for any other classification task. Analyses of the models' prediction results indicate that the incorrect

labeling of instances as no_relation class or negative relation contributes significantly to the noise in the data. Hence, this work proposes two different strategies for identifying potentially noisy negative relation instances for elimination and reannotation. The first strategy, the intrinsic strategy, is based on finding the nearest neighbor to the model's false negative prediction. Whereas, the second strategy, the extrinsic strategy, requires a subset of clean TACRED instances. Models trained on a dataset with elimination based on the intrinsic strategy show improvement when models are evaluated on the cleaner version of the test set. The performance of the models significantly improved with the extrinsic strategy for both the eliminated and reannotated datasets. Furthermore, identifying noisy instances among positive relation classes using the extrinsic strategy shows more improvement in models' performance.

In the previous chapter, we scanned through the entire dataset and identified a few hundred instances for reannotation. Whereas, in this chapter, the number rose to a few thousand i.e., around 4% - 10% of the dataset. The computation involved in identifying such a small percentage could be very expensive, crossing the reannotation budget, when the size of the dataset becomes large. This demands the need for a flexible-budget framework for reannotation. In the next chapter (Chapter 5), we work in this direction.

Chapter 5

Budget-Sensitive Reannotation of noisy RC dataset

Chapter Highlights

- We observe that the majority of the reannotation tasks, including the ones we proposed in previous chapters, consider a fixed set of instances for reannotation, i.e., either the entire dataset or a small subset based on heuristics or experts' suggestions.
- As a result, these approaches lack flexibility in the number of instances they reannotate.
- We propose the concept of a reannotation budget to incorporate flexibility in the number of instances to reannotate.
- We further explore four different strategies to identify the most noisy instances within a given reannotation budget.
- Our empirical results suggest that baseline models' performance on TACRED is inflated.
- We also observe improvement in models' performance when reannotating only a part of the training data.
- This chapter is based on the paper "Relation Classification Model Performance as a Function of Dataset Label Noise" submitted to ACM Journal of Data and Information Quality (ACM JDIQ).

Abstract

In the previous chapters, we have seen that the large datasets for relation classification are known to have data points with incorrect annotations, resulting in label noise. We have also shown that the performance of RC models improves with the reannotation or elimination of noisy instances from the dataset TACRED. This is also evident from the results of TACRev and ReTACRED, two cleaner variants of TACRED. All of these works, however, either re-annotate a small fixed portion of the data or the entire dataset. They don't have the freedom to choose how much data to reannotate and which instance to reannotate. To overcome these limitations, we address the last RC challenge highlighted in Chapter 1 and introduce the concept of a reannotation budget. This enables us to analyze the RC model's performance as a function of label noise.

The immediate follow-up question is: given a specific reannotation budget, which subset of data should we reannotate? For example, an incorrectly annotated instance from TACRED like *Ezra Randle, Oklahoma overseer for the Christ Holy Sanctified Church, said Daniels had been traveling to Anadarko for four or five years and 85 percent of the time, she made the trip alone.* labeled with relation *per:Age* between entities *Daniel* and *four* should be considered with high priority than correctly annotated instance like *The governor of Punjab province, Salmaan Taseer, where Bibi has been held in jail for more than a year, said he had forwarded a petition presenting the facts of the case to President Asif Ali Zardari on Monday.* labeled with *no_relation* between *Bibi* and *more than a year*. We explore four strategies to select data points for a given reannotation budget. Two of these strategies rely on the graph distance between actual and predicted relation labels, whereas, of the other two, one is random picking and the other is based on models' confidence score.

We design our experiments to answer three specific research questions. First, does our strategy select novel candidates for reannotation? Second, for a given reannotation budget, is our reannotation strategy more efficient at catching annotation errors? Third, what is the impact of data reannotation on RC model performance measurement? Our experimental results show that the reported performance of supervised models on noisy TACRED data is inflated. We also show that model performance improvement can be achieved by reannotating only a part of the training data. To the best of our knowledge, this is the first work that analyzes the relation between RC model performance improvement and the amount of data reannotation required for it.

5.1 Introduction

In the literature, the problem of learning from noisy datasets is widely popular, especially for computer vision related tasks. The proposed works can be broadly categorised as robust architecture [79], robust regularization [80], robust loss design [81], and sample selection [82]. These approaches can only improve the model performance to an extent where the model efficiently learns from clean data, ignoring or reducing the impact of noisy datasets. Thus, the underlying dataset is still noisy and will continue to contribute to the model's learning. This is probably one of the reasons for the RC model's performance being throttled in the range of 75% to 80%.

The central question that this chapter address is: how does the performance of a supervised model vary with respect to the label noise in the training dataset? Answering this question is crucial for many real-world applications for two main reasons. First, most of the datasets used for training large supervised models such as Deep Neural Networks (DNNs) are crowd-sourced. Such datasets are known to have significant label noise due to multiple factors, such as the complexity of the annotation task and low wages for crowd-sourced annotators. Second, these crowd-sourced datasets are large. Reannotating such large datasets is a time-consuming and costly process. Our work aims to understand the relationship between the cost of data reannotation and the performance of supervised models. Understanding this relationship can be helpful while planning for data reannotation.

Existing methods for RC dataset reannotation follow two extremes. They either select only a tiny fraction for reannotation [6] or go for reannotation of the complete data [8]. Both these processes are extremely time-consuming and costly. In contrast, our approach provides flexibility in terms of the reannotation budget (Figure 5.1), which is the number of sentences we can afford to reannotate. Given a large, noisy, labeled dataset and a limited budget for reannotation, which data points should we prioritize for reannotation? If our reannotation budget permits us to reannotate a significant part of the data, then the strategy for selecting data points for reannotation does not matter. However, in the real world, the reannotation budget is far smaller as compared to the dataset size for two reasons. First, reannotation is a costly and time-consuming task. Second, avoid making redundant efforts to reannotate a correctly labeled data point.

In the context of the RC task, this chapter proposes two strategies for selecting instances for reannotation. Our approach capitalizes on the taxonomic hierarchy of relation labels, which is largely an ignored aspect of the RC datasets. For each instance, we compute the graph distance between the actual label provided in the dataset and the predicted label using

Fig. 5.1 A dataset with noisy labels is sorted to prioritise clearly mislabelled samples, maximising the number of corrected samples given a fixed relabelling budget.

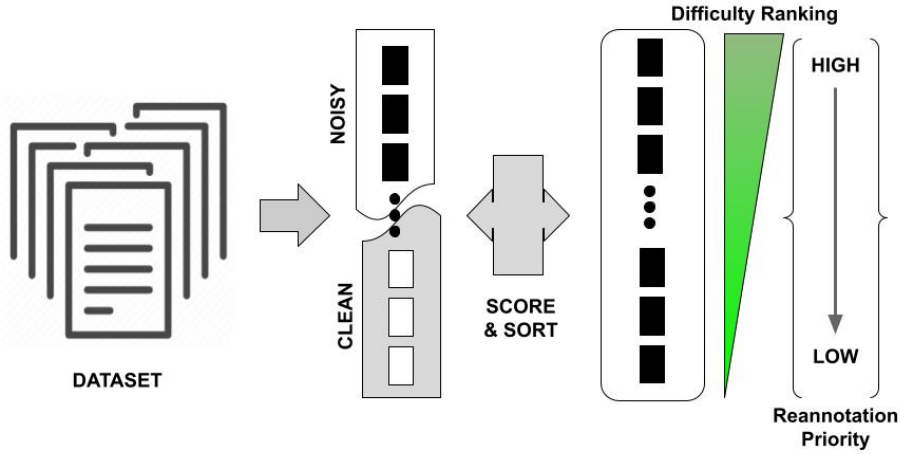
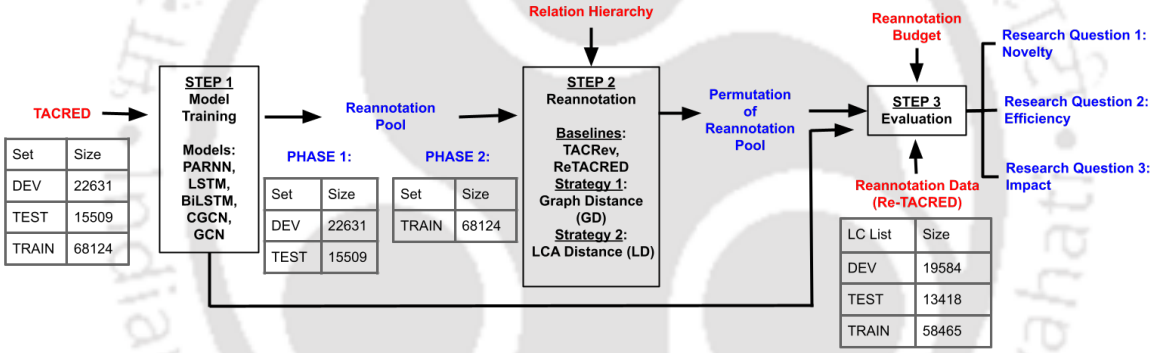


Fig. 5.2 Overview of our work



an ensemble of RC models. Data points with a higher value for this distance are given higher priority for the reannotation task. We experimentally evaluate our reannotation strategies on the well-known TACRED dataset[7] along with confidence-based selection and random selection strategies. An overview of our work is presented in Figure 5.2.

Our research contributions in this work can be summarized as follows:

- We introduce the concept of reannotation budget to provide flexibility about what fraction of dataset to reannotate.
- This is the first work that uses relation label hierarchy while selecting data points for reannotation for the RC task.
- We perform extensive experiments using the popular RC dataset TACRED. We show that our reannotation strategies are novel and more efficient when compared with the existing approaches.

- Our experiments suggest that the reported performance of existing RC models on the noisy dataset is inflated. The F1 score of these models drops from the range of 60%-70% to as low as below 50% when tested on clean test data generated using our reannotation strategy.
- We also observe that significant improvement can be achieved in the RC model performance by reannotating only a part of training data.

All the experimental results are specific to the RC task. However, our reannotation strategy can be applied to any task and dataset where the label hierarchy is available. The rest of the chapter is organized as follows. We review the related work in Section 5.2. Our reannotation approach along with experimental results is presented in Section 5.3. In Section 5.4, we have discussed impact of our work on baseline models' performance. Limitations of our work are discussed in Section 5.5. Finally, we conclude in the Section 5.6.

5.2 Related Works

Some of the earliest methods for the RC task were based on pattern extraction. They [83, 84] use syntactic patterns to construct rules for extracting relational facts from texts. To reduce the human effort in identifying relation facts, statistical relational relation extraction has been extensively explored in two directions, namely, *feature-based methods* [40, 33, 34] and *kernel-based methods* [36–39]. Following the success of embedding models in other NLP challenges, low-dimensional textual embeddings are also used for the RC task.

Neural Relation Extraction models introduce neural networks for capturing relational facts within text, such as *recursive neural networks* [41, 85], *convolutional neural networks (CNN)* [42, 43, 18], *recurrent neural networks (RNN)* [7, 44], and *attention-based neural networks* [45, 46]. Recently, *Transformer* [19] and *pre-trained language models* [24] have also been explored for relation extraction [48, 58, 9, 49, 51, 50] and have achieved new state-of-the-arts performance.

Out of all the RC datasets [27, 29, 26, 7, 56], TACRED [7] is the largest and most widely used dataset. It contains more than 100 thousand sentences for 42 relation classes (Train: 68124 sentences, Dev: 22631 sentences, and Test: 15509 sentences). Each sentence contains a pair of real-world entities representing subject and object entities and is annotated with either one of the 41 relations or `no_relation`. Sentences are manually annotated with relation labels using the Amazon Mechanical Turk crowdsourcing platform, where each

annotator is provided with a sentence, subject and object entities, and a set of relations to choose from.

A recent study by Alt et. al. (referred to as TACRev) trained 49 different RC models and selected the 5000 most miss-classified instances from TEST and DEV partitions for reannotation [6]. With the help of their expert annotators (trained linguists), they ended up modifying 960 out of 1923 sentences from TEST and 1610 out of 3088 sentences from DEV. The revised dataset resulted in an average 8% improvement in the F1-score, suggesting the noisy nature of the TACRED dataset. However, their work has two main bottlenecks. First, their method has a fixed set of sentences to reannotate. Second, they use only model confidence to select sentences for reannotation. We compare our strategies against TACRev by extending their method for the reannotation budget. To simulate the TACRev reannotation strategy for a given reannotation budget, we select the top sentences from the dataset where RC models have the highest confidence.

Another recent study by Stoica et. al. (referred to as ReTACRED) reannotated the complete TACRED dataset using crowd-sourcing [8]. For more effective crowd-sourcing, they have made slight modifications to the original TACRED relation list. They corrected 3936, 5326, and 13923 annotation errors in the TEST, DEV, and TRAIN partitions, respectively. They have also eliminated 2091, 3047, and 9659 sentences from the TEST, DEV, and TRAIN sets, respectively, for various reasons. We use the data from their reannotation experiment to simulate our reannotation strategies. Our work cannot be directly compared against ReTACRED as they simply use the brute-force method of complete reannotation. However, for a meaningful comparison with ReTACRED, we extend their approach by selecting a random set of sentences for a given reannotation budget.

The main takeaway points from the existing work can be summarized as follows:

- Deep Learning is the paradigm of SOTA RC models. Hence, we will focus only on Deep Learning based models for our work. Even though we have used simpler models for this work due to resource constraint, similar results can be achieved using complex neural models.
- TACRED is the RC dataset used by SOTA models. Also, the reannotation data for the complete TACRED dataset is available. Hence, for our experiments we use the TACRED dataset.
- Both the existing works on RC dataset reannotation are rigid. They fail to provide flexibility about how much data to reannotate. Hence, we introduce the concept of reannotation budget to overcome this limitation.

- The concept of taxonomic hierarchy of relation labels is not yet used for the RC dataset reannotation. In this work, we use taxonomical relation hierarchy for TACRED relation labels (from Chapter 3) for calculating the shortest-path distance between ground-truth and prediction.

5.3 Proposed Work

Considering the insights from related work, our goal is to progressively reduce the label noise through the budget-sensitive reannotation method and observe its effect on the TACRED dataset and deep learning models. Consider an RC dataset D. Let N be the set of noisy or mislabeled sentences in D. As N is a subset of D, $|N| \leq |D|$. Let R represent the set of sentences that we are going to reannotate from D. Our reannotation budget is $|R|$. In other words, we can afford to reannotate only $|R|$ sentences out of $|D|$. The goal of any reannotation method should be to maximize the $|N \cap R|$. In the ideal case, N and R should be identical sets.

Given a reannotation budget $|R|$, the immediate follow-up problem is: what subset of the data should we reannotate? If our reannotation budget permits us to reannotate a significant part of the data ($|R| \approx |D|$) then the strategy for selecting data points for reannotation does not matter. However, in the real world, the reannotation budget is far smaller compared to the dataset size ($|R| \ll |D|$) for two reasons. First, reannotation is a costly and time-consuming task. Second, redundant efforts in reannotating a correctly labeled data point should be avoided. In the context of the RC task, this work proposes two strategies for selecting data points for reannotation. Our approach capitalizes on the taxonomic hierarchy of relation labels, which is largely an ignored aspect of the RC datasets. For each data point, we compute the average shortest-path distance between the actual label provided in the dataset and the predicted label using an ensemble of RC models. Data points with a higher value for this distance are given higher priority for the reannotation task. An overview of our work is presented in Figure 5.2. Our work is divided into three steps: model training, reannotation, and evaluation. These steps are described in the following subsections.

5.3.1 Model Training

The first step trains multiple RC models using the original TACRED dataset. However, we transform the relation labels to match the ReTACRED labels. This transformation is necessary as we have the reannotation data available from ReTACRED only. For our experiments

Model	TACRED			TACRev			ReTACRED		
	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score
PALSTM	67.8	64.6	66.2	73.7	74.8	74.3	78.8	80.1	79.4
LSTM	65.4	61.6	63.4	70.9	71.1	71	80.3	76.3	78.2
BiLSTM	65.2	60.4	62.7	70.1	69.2	69.6	78.9	77.4	78.1
CGCN	71.2	62.6	66.7	77.5	72.6	75	80.5	80	80.2
GCN	69.2	60.3	64.5	73.8	68.5	71	79.7	76.9	78.3
CNN	64.3	59.7	61.9	69.5	68.8	69.1	75.4	77.6	76.5

Table 5.1 Baseline model performance on TACRED, TACRev, and ReTACRED. All the reported results are based on our implementation of the available code.

we have selected following six Deep Learning based RC models: PALSTM, LSTM, and Bi-LSTM following [7][1], CGCN, and GCN following [57][2], and CNN following [18][3]. We have trained these models using the same set of hyper-parameters as mentioned in their original papers (model performance of our implementation on all three datasets are reported in Table 5.1). This step can be further improved by training more RC models. However, we were limited by the available computing infrastructure.

All the input word vectors are initialised using pre-trained Glove vectors [12]. For training **GCN** and **CGCN**, we used the hyper-parameters following [57]. We used LSTM hidden size and feedforward hidden size as 200. We used two GCN layers and two feedforward layers, SGD as an optimizer, initial learning rate of 1.0 which is reduced by a factor of 0.9 after epoch five every time dev set performance plateaus. We trained the model for 100 epochs. We used word dropout of 0.04 and dropout of 0.5 to LSTM layers. All other embeddings (such as NER, POS) size is fixed as 30.

For training **PALSTM**, **LSTM**, and **BiLSTM**, we followed [7] for hyper-parameters. We have used two-layer stacked LSTM layers for all the models with a hidden size of 200. We used *AdaGrad* with a learning rate of 1.0 which is reduced by a factor of 0.9 after the 20th epoch. We have trained the model for 30 epochs. We used word dropout of 0.04 and dropout of 0.5 to LSTM layers. All other embeddings (such as NER, POS) size is fixed as 30.

For training **CNN** also, we used parameters used by [7]. We used *AdaGrad* with learning rate of 0.1 and trained model for 50 epochs. For CNN, we use 500 filters of sizes $[2,3,4,5]$ and apply *l2-regularization* with a co-efficient of 0.001 to all filter weights.

[1] https://github.com/yuhaozhang/tacred-relation
[2] https://github.com/qipeng/gcn-over-pruned-trees
[3] https://github.com/roomylee/cnn-relation-extraction

5.3.2 Reannotation

The second step simulates the reannotation process by creating a specific permutation of the data points in the reannotation pool. We have divided our experiments into two phases. For Phase 1, we have considered the TEST and DEV partitions of the TACRED dataset as the reannotation pools. For Phase 2, we have considered TRAIN partition for reannotation. For Phase 2, we assume that the TEST and DEV partitions are already reannotated. In our simulation, the crowd-sourced workers will reannotate only a subset of data points from the reannotation pool, depending on the reannotation budget. We have experimented with four reannotation strategies: two from the literature (TACRev[6] and ReTACRED[8]) and our own proposed two strategies. Given a particular budget for reannotation, we have to select a set of sentences from the reannotation pool to perform the reannotation task. A reannotation strategy creates a ranked list of the reannotation pool. The goal of a reannotation strategy is to ensure that sentences with annotation errors appear at the top of the ranked list. Such a ranking is expected to use a given reannotation budget effectively. Reannotating all the sentences in the reannotation pool will require a large number of resources. In the real world, we can typically afford to reannotate only a fraction of the reannotation pool. We expect that a good reannotation strategy should be able to catch most of the annotation errors with a relatively small reannotation budget.

A trained RC model, when presented with a data point for relation classification, assigns a relation label along with the confidence score. This score indicates how confident the model is while assigning the relation label. The TACRev strategy uses multiple RC models for ranking the data points based on the *confidence* of the RC models. In our experiments, we used six RC models mentioned in the subsection 5.3.1. The ReTACRED method does not have the concept of ranking data points. It is a brute-force strategy that simply reannotates the whole reannotation pool. To simulate the ReTACRED strategy with the reannotation budget, we simply *pick up random sentences* from the reannotation pool to perform the reannotation task.

Our reannotation strategy is based on the taxonomic-hierarchy prepared by Parekh et al. [10]. Following their work, we arranged all relations from the TACRED dataset into a taxonomic hierarchy. The relation labels are arranged as a tree. Each relation label has a parent and multiple or zero children. Please refer to Chapter 3.4 for the TACRED relation hierarchy. With such a hierarchy, it becomes feasible to measure the error in the model's prediction. Both our reannotation strategies utilize this hierarchy for selecting candidates for reannotation.

5.3.2.1 Graph Distance Strategy

Consider the following sentence, S, which has its relation label annotated in the dataset as $AR(S)$. AR stands for Actual Relation. We have a list of K different RC models: M_1 to M_K. Each model M_i assigns the sentence S a relation label $PR_i(S)$. PR stands for "Predicted Relation." Both $AR(S)$ and $PR_i(S)$ can be located as nodes in the relation hierarchy tree. Our first reannotation strategy measures the length of the path between these two nodes in the tree. We refer to this strategy as the "Graph Distance" (GD) strategy. For each sentence S, we compute the following score:

$GD(S) = (\sum_{i=1}^{K} f_i(S))/K$,where

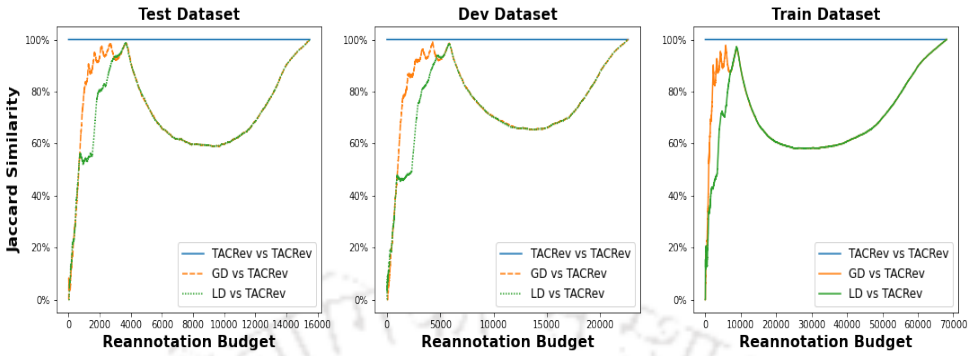
$f_i(S) = Distance(AR(S), PR_i(S))$

The *Distance* function computes the length of the shortest path between two tree nodes. This is a fundamental problem in graph algorithms, and it can be solved in O(h) time, where h is the height of the tree.Assume that $AR(S)$vis $per : parent$ and $PR(S)$ is $per : age$ for a given sentence S.In this case, the shortest path between these two labels is ($per : parent$, $per : family$, $per - per$, per, $per - misc$, $per : age$) with the *Distance* of five edges. The function $f_i(S)$ measures the disagreement between the prediction of model M_i and the label given in the dataset. A high $f_i(s)$ value indicates a high level of disagreement between the model M_i and the currently assigned label in the dataset. The $GD(S)$ score is the average of all K models. Our Graph Distance reannotation strategy considers sentences with a high GD score as preferred candidates for reannotation. Intuitively, we are selecting sentences where multiple RC models have strong disagreement with the currently assigned label in the dataset.

5.3.2.2 LCA Distance Startegy

Our second strategy involves fine-tuning the computation of disagreement between model prediction and currently assigned labels in the dataset. We locate both $AR(S)$ and $PR_i(S)$ in the relation hierarchy and find their Lowest Common Ancestor (LCA). The LCA computation in the tree is also a basic graph problem that can be solved in time complexity $O(h)$, where h is the height of the tree. Consider the same example with $AR(s)$ as $per : parent$ and $PR(S)$ as $per : age$. In this case, the LCA of these two labels is the node per. Consider the path from the root of the relation hierarchy to the node $AR(S)$. The LCA node is the point at which the model prediction starts to differ from the currently assigned label in the dataset. We compute the following score for each sentence S:

Fig. 5.3 Similarity of a reannotation strategy with TACRev.



$LD(S) = (\sum_{i=1}^{K} g_i(S))/K$,where

$g_i(S) = Distance(AR(S), LCA(AR(S), PR_i(S)))$

The function $g_i(S)$ fine-tunes the definition of disagreement. Instead of considering the complete path between $AR(S)$ and $PR_i(S)$, we are now considering only the part up to the LCA of these two nodes. We call this reannotation strategy as LCA Distance (LD) strategy. Each reannotation strategy creates a permutation of the reannotation pool by sorting sentences in the descending order of the corresponding scores.

5.3.3 Evaluation

After step 2, each reannotation strategy will create its own permutation of the reannotation pool. Based on the reannotation budget $|R|$, we will select the top $|R|$ data points from each permutation. While comparing our strategies with those of TACRev and ReTACRED, we need to answer three important questions. *First* about **novelty**: do our strategies provide novel candidates for reannotation? *Second* about **efficiency**: do our strategies catch more annotation errors for a given reannotation budget? And *third* about **impact**: what is the effect of reannotation on the model performance?

5.3.3.1 Novelty in Candidate Selection

The novelty of GD and LD against ReTACRED is trivial because ReTACRED selects random data points for reannotation. Please refer to Figure 5.3 for the analysis of novelty against TACRev. The X-axis indicates the reannotation budget. The Y-axis indicates the *Jaccard similarity*, that is the size of the intersection divided by the size of the union. TACRev's

similarity to itself is always 100%. We can observe that for a low reannotation budget, the similarity score grows quickly and approaches 100%. This indicates that initially, all strategies choose similar candidates for reannotation. For medium budget values, the similarity falls significantly. This indicates that our strategies differ significantly from TACRev for medium-budget value. For a higher reannotation budget, the similarity score again approaches 100% as expected. When the reannotation budget is high, we will reannotate almost the entire dataset. In such a scenario, there is hardly any chance for novel candidate selection.

The qualitative analysis of novelty provided by all three strategies is presented in the Table 5.2. It can be observed that all the strategies selects different samples.

S.no	TACRev based Selection		GD based Selection		LD based Selection	
	Sentence	Avg. confidence	Sentence	Avg. GD	Sentence	Avg. LD
1	Ezra Randle, Oklahoma overseer for the Christ Holy Sanctified Church, said Daniels had been traveling to Anadarko for four or five years and 85 percent of the time, she made the trip alone. **relation** - *per:age*	0.999167	The ADF is a Muslim rebel group that claimed to fight for equal rights for Muslims in Uganda but was driven out of the country in 2001 and has since been based in eastern Democratic Republic of Congo. **relation** - *org:country_of_headquarter*	6	Supporters like Ble Goude have branded the Golf Hotel a rebel base, and both FDS troops and civilian protesters have begun to harass UN patrols in Abidjan, which is still firmly under the control of Gbagbo's forces. **relation** - *per:cities_of_residences*	5
2	He formed his own company, TAU, for a while, sold part to Trimble, left Trimble to start a GPS business line at Motorola (where he was also a VP), from there to Rand McNally in Chicago, and more recently to NavTeq. **relation** - *per:sibbling*	0.997833	Then in May, Samudio went back to Rio to find Souza, presumably, the police said, to prove to him the baby was his. **relation** - *per:cities_of_residences*	6	"The United States would view favorably the release of Alan Gross so that he can return to his family," she added. **relation** - *per:countries_of_residence*	5
3	The next day, Halliburton told her that if she left Iraq to get medical treatment, she could lose her job. **relation** - *per:other_family*	0.997833	In October, she filed a complaint with the police in Rio saying he had kidnapped her and tried to threaten her into having an abortion. **relation** - *per:other_family*	6	Jamie Leigh Jones is testifying on Capitol Hill this afternoon. **relation** - *per:cities_of_residence*	5
4	The market traded nervously, jerking the Dow Jones industrial average above and below the 13,000 mark throughout the day as investors wrestled with reports about potential trouble at Countrywide Financial Corp. and KKR Financial Holdings LLC. **relation** - *org:alternate_names*	0.997667	Gaunt and haggard, freed Italian aid worker Eugenio Vagni says his release after being held for six months by armed Islamic militants in the Philippines was a day he thought might never come. **relation** - *per:countries_of_residences*	6	Last month, Norris Mailer published her memoir, "A Ticket to the Circus," in which she recounted the ultimatum she gave to her husband in 1991 after finding notes from Mallory and other girlfriends in the author 's studio. **relation** - *per:spouse*	5

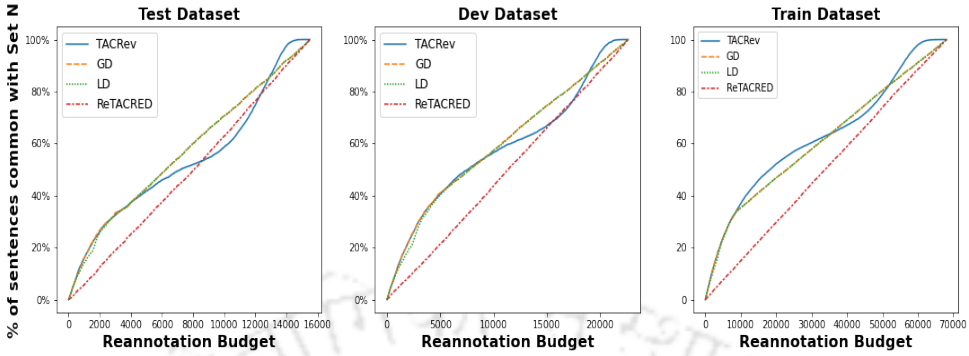
Table 5.2 First four sample instances selected by three different strategies. Avg. is the average of score returned by all models used. Samples are selected from the list of sentences. subject entity is in red color and object entity is in green color.

5.3.3.2 Efficiency

ReTACRED performed a complete reannotation of TACRED data [8]. They observed that the set of noisy data points (N) in TACRED are: 5326 sentences from DEV, 3936 sentences from TEST, and 13923 sentences from TRAIN. To evaluate the performance of our reannotation strategies, the obvious way is to carry out the reannotation task again on our own. However, we chose to simulate the performance of our reannotation strategies using the set N for two main reasons. First, the set N is meticulously prepared by the ReTACRED authors, who devised a large-scale, multi-stage crowdsourcing experiment.Second, we did not have the financial budget available to carry out the large-scale reannotation task on our own. We took the labels from ReTACRED as the gold standard labels.

Please refer to Figure 5.4. The X-axis indicates the reannotation budget. The Y-axis depicts the percentage of sentences from the set N that we can find in the given budget using a particular reannotation strategy. For example, when we reannotate the top 1000 sentences

Fig. 5.4 Annotation errors corrected by various reannotation strategies.



from the TEST partition using the LD strategy, it contains 566 sentences, around 14.4 percent of the set N. We can observe that for a low reannotation budget, GD, LD, and TACRev have the same efficiency. This is expected as their similarity was close to 100% for the low reannotation budget. However, for the medium reannotation budget, our strategies outperform TACRev for the TEST and DEV partitions. While reannotating the TRAIN partition, for medium budgets, there are two distinct trends. Initially, for the reannotation budget range of 10,000 to 35,000 sentences, TACRev outperforms both GD and LD. However, later, for the range of 35,000 to 55,000 sentence reannotation budget, GD and LD perform slightly better than TACRev. For the very high value of the reannotation budget, TACRev performs slightly better than our strategies for all three partitions of the data. The efficiency of GD and LD strategies is almost the same as their lines are overlapping in the figure. The efficiency of ReTACRED grows linearly as it chooses random sentences for reannotation.

5.4 Impact on Model Performance Measurement

While measuring the impact of data reannotation on the model performance, we will analyze the two phases separately. For Phase 1, we train our models only once, as we are not reannotating our TRAIN partition during this phase. For Phase 2, after each reannotation budget, we have to retrain our models to evaluate the effect of a partial reduction in the label noise in the TRAIN partition.

5.4.1 Phase 1

Please refer to Figure 5.5. The X-axis indicates the reannotation budget. The Y-axis indicates the F1 score of each model. Please note that we are only reannotating the TEST data here. The models are trained on the noisy TACRED dataset. We can observe that for three reannotation strategies (TACRev, GD, and LD), initially, the model performance improves with reannotation. This indicates that RC models agree with the initial reannotation. However, the performance of all models falls significantly with a higher reannotation budget. The test dataset is the cleanest when the reannotation budget is set to maximum. The F1 score for all models at this point indicates that the original performance reported in the literature for these models was inflated. The main reason for this inflated value is the significant noise in the TEST partition of the TACRED dataset.

None of the previous works could catch this trend for the following reason. We have conducted experiments here with noisy training data and varying degrees of noise reduction in the test data. However, previous works can be classified into the following categories:

Noisy training and noisy test data: Several works that directly used TACRED data are training their RC models on TACRED's noisy TRAIN partition [57]. They are also testing their RC models on noisy TEST data. As a result, their reported F1 score is in the range of 60% to 70%. This corresponds to reannotation budget zero in Figure 5.5.

Noisy training and partially clean test data: The TACRev study reported an average improvement of 8% in the F1 score with the reannotation budget of about 2300 sentences from the TEST partition [6]. Our results for TACRev in Figure 5.5 are consistent with their reported numbers. We can observe that while using the TACRev reannotation strategy, the performance of all models peaks with a reannotation budget of around 2300. However, their results are a special case of our experiments with a fixed reannotation budget of 2300. We can observe a similar trend for our reannotation strategies GD and LD in Figure 5.5. Further cleaning of the TEST dataset beyond the budget of 2300 actually results in a performance drop for all RC models and all reannotation strategies.

Completely clean training and completely clean test data: ReTACRED performed a complete reannotation of all three partitions of the TACRED dataset [8]. As a result, they report at least a 10% increase in F1 scores for various RC models when compared to previous work.This case cannot be captured in Figure 5.5 because we are not making any changes to the TRAIN partition of the TACRED dataset.

In Figure 5.5, we can observe that for the ReTACRED strategy, the model performance goes on monotonically decreasing with the reannotation budget. This is expected as the order of the sentences for reannotation is random. From the other three reannotation strategies, we can observe that reannotation of only about the top 2500 sentences helps RC models boost their performance. All these 2500 sentences are ordered randomly in the ReTACRED strategy. Reannotation of the rest of the sentences from the TEST partition does not agree with the RC model prediction and brings down the model performance.

5.4.2 Phase 2

Please refer to Figure 5.6. The X-axis indicates the reannotation budget. The Y-axis indicates the F1 score of each model. For Phase 2, we start with a clean TEST and a noisy TRAIN. We will progressively go on reducing the label noise by increasing the reannotation budget. We can observe that for all reannotation strategies, the initial improvement in the model performance is slow. However, after a certain threshold, there is a quick improvement in the model performance within a short range of the reannotation budget. This threshold is different for various strategies (GD:40%, LD:40%, TACRev:60%, and ReTACRED:25%). For all models, the final F1 score with complete reannotation is in the range of 75% to 80%. However, most of the model performance improvements can be achieved by reannotating only a fraction of the TRAIN partition. For example, RC model performance improves from the range of 50%-55% to the range of 70%-75% with reannotation of about only 60%-70% of the TRAIN partition.

5.5 Limitations of Our Work

The first limitation of our work is that we trust the annotations from ReTACRED as the ground truth. In reality, the ReTACRED annotations are also done using crowd-sourcing. However, as compared to the original TACRED dataset, they have taken more precautions to avoid annotation errors. For example, they have rearranged the relation labels to avoid confusion in labeling. They have also created better instructions for the crowd-sourced workers. One way to further reduce the possibility of errors in the data is to get annotations from domain experts. However, considering the scale of the dataset, it is a challenging task.

The second limitation of our work is that it requires a taxonomic hierarchy of relation labels. We have created the relation hierarchy manually for the TACRED dataset. It would be helpful if such a hierarchy could be created automatically from the RC dataset.

5.6 Chapter Summary

In this work, we analyzed the RC model performance as a function of label noise in the dataset. The reported F1 score of various RC models in the literature is in the range of 60% to 70%. These works report the inflated performance of RC models due to label noise in the TEST partition. However, at the end of Phase 1 evaluation, we realize that their actual F1 score is in the range of 45% to 55%. This performance can be improved by reducing the label noise in the TRAIN partition. Based on the Phase 2 evaluation, we can achieve significant model performance improvements by reannotating only a portion of the data.

We also introduced the concept of a reannotation budget to provide flexibility about how much data to reannotate. In this chapter, we proposed two reannotation strategies and compared them against two existing reannotation strategies. Our reannotation strategies (GD and LD) are novel and efficient.

A high rate of reannotation indicates that crowd-sourced labels are not trustworthy, especially for complex NLP tasks such as relation classification. Hence, data annotation should not be considered a one-time task. Rather, it should be treated as a budget-sensitive and iterative process where data is labeled in multiple iterations.

Fig. 5.5 Performance of RC models trained on noisy training set after reannotation of test data at different reannotation budget.

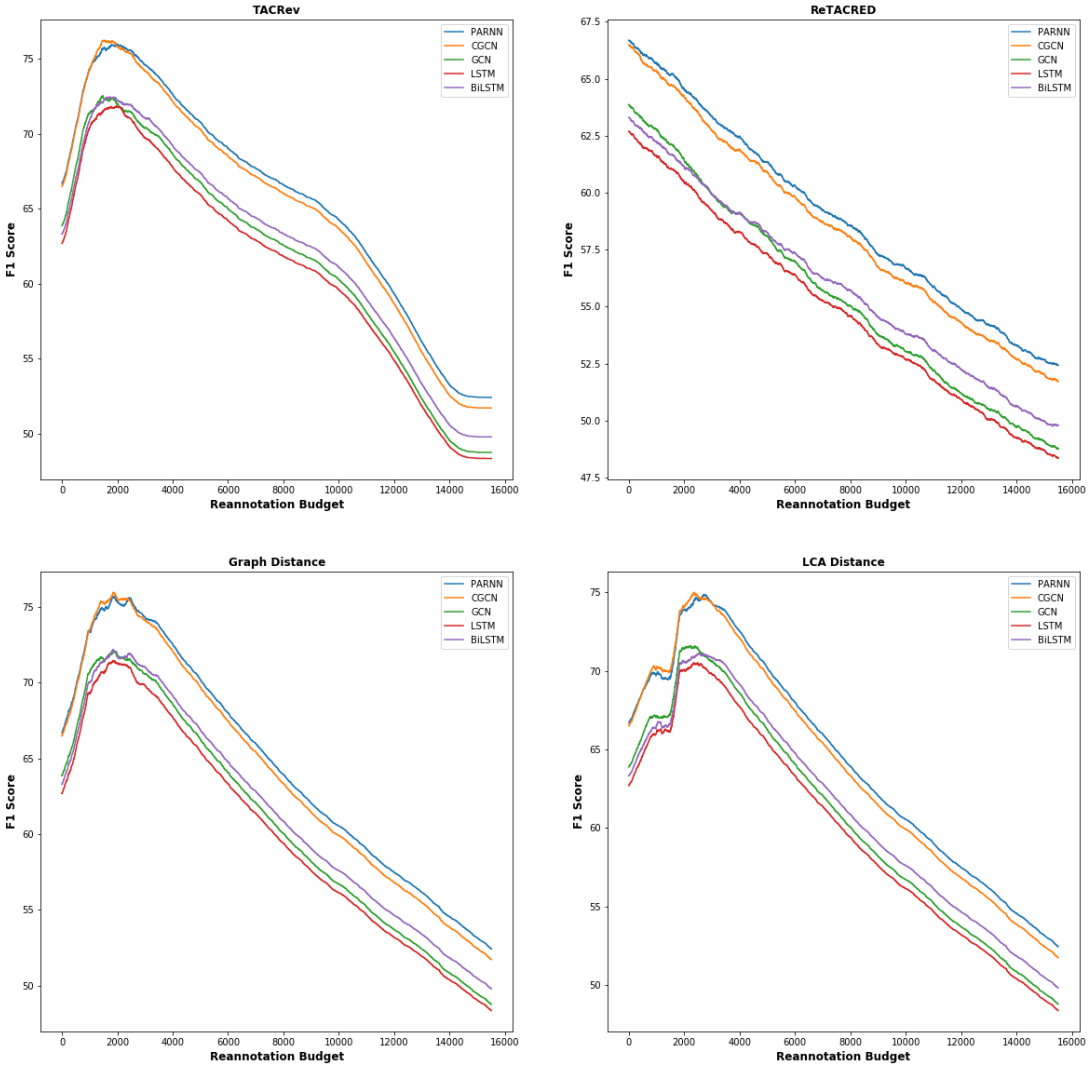
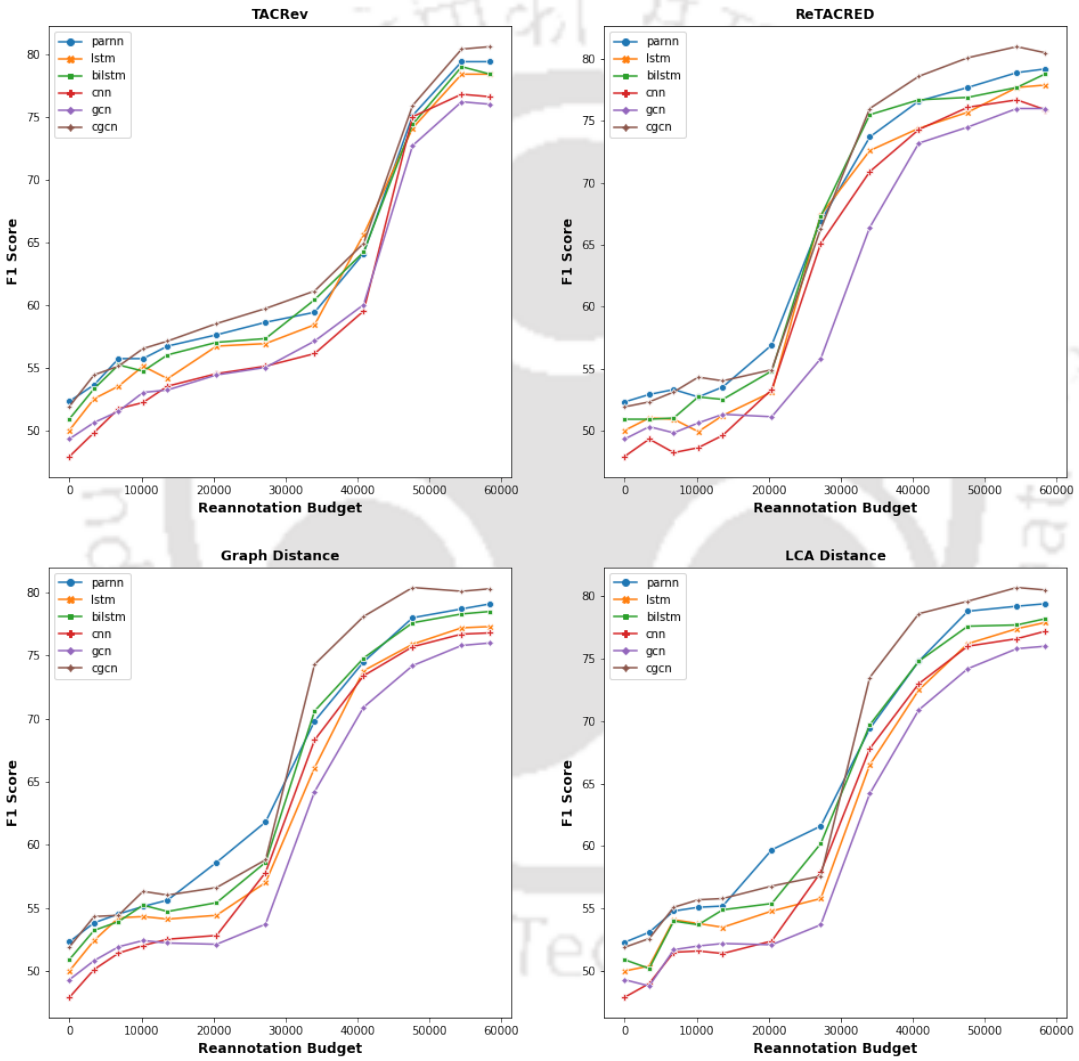


Fig. 5.6 Performance of RC models after reannotation of training data at different reannotation budget on clean test data.



Chapter 6

Conclusions and Future Work

In this thesis, we proposed heuristics and strategies to address the challenges associated with the large RC dataset. We performed all our experiments using TACRED. We addressed three major issues: a lack of relation labels, a lack of relation hierarchy, and a noisy and imbalanced dataset. We addressed all these challenges, mainly focusing on reducing noise from the RC dataset with time-efficient and cost-effective approaches.

In Chapter 3, exploring Multiple KBs, we curated a list of 623 relations and arranged them in a taxonomical relation hierarchy. Further, we used that relation hierarchy as a template for creating a similar relation hierarchy for TACRED relation labels. We used the generated TRH to eliminate ambiguous instances, as well as to deal with the long-tail distribution of relation labels and ambiguity in relation boundaries, we relabeled some finer relations with their coarser relation labels following relation hierarchy. Finally, we used hierarchical distance to scale cross-entropy loss (CE Loss) and proposed *hierarchical distance scaled cross-entropy loss (HCE Loss)*.

Following TRH in our previous chapter (Chapter 3), we could only eliminate a few noisy positive instances. Consequently, the true picture of noise in the TACRED dataset was still unclear. Therefore, in Chapter 4, we explored the model-based characterization of noise present in the TACRED dataset and two strategies to handle potentially noisy instances. Analyses of model prediction results indicated that the incorrect labeling of instances as *NO_RELATION* class or "negative relation" is predominantly responsible for the noise in the data. Hence, we proposed two different strategies for identifying potentially noisy negative relation instances for elimination and reannotation. The first strategy, **intrinsic strategy**, is based on finding the nearest neighbor to the model's false negative prediction. Whereas,

the second strategy, **extrinsic strategy**, requires a subset of clean TACRED instances. Our experimental results show that elimination and reannotation based on an extrinsic strategy give better datasets in comparison to the intrinsic strategy. Model performance improved further when we reduced noisy positive instances, also using an extrinsic strategy.

In Chapter 5, we introduced the concept of a reannotation budget to provide flexibility about how much data to reannotate. We also proposed two strategies based on the taxonomic hierarchy of relation labels (from our earlier work, Chapter 3) for selecting data points for reannotation. We have shown that our reannotation strategies are novel and more efficient when compared with the existing approaches. Our experiments suggest that the reported performance of existing RC models on the noisy dataset TACRED is inflated. Furthermore, we show that significant model performance improvement can be achieved by reannotating only a part of the training data.

Future Work Directions

While the dissertation has made significant progress in the advancement of relation classification and learning from noisy RC dataset, there are still several open problems that remain unaddressed. Many of which are worth pursuing as future work, as discussed:

Automated Expansion of Taxonomcal Relation Hierarchy for multiple knowledge bases:

Nakashole et al. [84] created a hierarchy for OpenIE relational phrases. Adapting such strategies along with efficient hypernymy detection algorithms and ontology alignment approaches can become the starting point of research in this direction. Moreover, in this work we considered only *person*, *organization*, and *location* entity types for the construction of TRH. However, this can be extended to any number of entity types. For example, by including *date*, relations such as `dateOfBirth`, `foundedinYear` can be included.

Efficiently Generating a large RE/RC Dataset:

RE/RC datasets are restricted because of limited number of relation labels considered in the benchmark datasets. This work provides a comprehensive list of 623 relations. This list,

along with the relation hierarchy, can further be used as a set of model relations for creating a sizeable sentence-level dataset for RE. On the other hand, using relation hierarchy, relation templates, and instances of relation triples from KBs, existing datasets can be improved with high confidence statements using generative models.

Learning from Noisy Datasets:

Even though the proposed methodologies led to a performance improvement, they were able to identify only a small subset of noisy instances. Moreover, identifying all noisy instances without manual intervention is challenging. Therefore, this work can further be extended by exploring approaches such as curriculum learning [70], multi-network learning [86], and label refurbishment [87]. Another interesting future direction could be to explore the idea of using uncertainty quantification methods [78] to identify potential noisy instances and utilize them for reannotation.

Resources and Github Repositories from this Thesis

Taxonomical Relation Hierarchy

Pre-pocessed 5 Million DBpedia Triples

Pre-pocessed 15 Million Wikidata Triples

Noise in TACRED: Characterization & Reduction (Github repository)

Budget-Sensitive reannotation (Github repository)

Publications

From Thesis

Manuscript Published

Conference

1. **Parekh, A.**, Anand, A. and Awekar, A., 2020. Taxonomical hierarchy of canonicalized relations from multiple Knowledge Bases. In Proceedings of the 7th ACM IKDD CoDS and 25th COMAD 2020.

2. **Parekh, A.**, Anand, A. and Awekar, A., 2022. Improving Relation Classification using Relation Hierarchy. In Proceedings of The 27th International Conference on Natural Language & Information Systems.

Manuscript Under Review

Journal

1. **Parekh, A**., Baranwal M., Dalal S., Anand, A. and Awekar, A., 2021. Noise in Relation Classification Dataset TACRED: Characterization and Reduction. The ACM Transactions on Knowledge Discovery from Data (TKDD).

2. **Parekh, A.**, Anand A. and Awekar A., 2022. Relation Classification Model Performance as a Function of Label Noise in the Data. The ACM Journal of Data and Information Quality (JDIQ) Special Issue on Deep Learning for Data Quality.

Outside Thesis

Manuscript Published

Workshop

1. Abhay M Shalghar, Ayush Kumar, Shobha G, Balaji Ganesan, Aswin Kannan, **Akshay Parekh**, 2022. Document Structure aware Relational Graph Convolutional Networks for Ontology Population. In Proceedings of ICLR Deep Learning on Graphs for Natural Language Processing.

Manuscript Under Review

Patent

1. Balaji Ganesan, Riddhiman Dasgupta, **Akshay Parekh**, Hima Patel, Berthold Reinwald. Neural-Based Ontology Generation and Refinement. US Patent Application No.: 16/937641.

Brief Biography of the Author

Akshay joined Ph.D. program at Department of Computer Science and Engineering of Indian Institute of Technology Guwahati, in July 2016. Before joining the Ph. D., he completed his Bachelor in Engineering (B.E.) from G. H. Raisoni College of Engineering, Nagpur, majoring in Computer Science and Engineering in the year 2013. Past that, he completed his Diploma in System Software Development (DSSD) from Center for Development of Advanced Computing Hyderabad (CDAC Hyderabad) in 2014. In summer 2019, the author did his research internship at IBM Research Lab India. He also worked as NLP Research Consultant for a startup ClearFeed.ai between October 2021 and January 2022. He has also participated and presented talks/tutorials at various events such as IRIS 2020 at IIT Gandhinagar, Amazon Research Day (2020, 2021), Nvidia GTC Developer's Program (2020, 2021), AI in Healthcare at IIT Guwahati (2019), Management Development Programme on AI in its possible implementation in Indian context, IIT Guwahati (2019) and many more. He was awarded travel grants for IRIS 2020 and CoDS-COMAD 2020 to present his work. His research interest includes Information Extraction, Deep Learning, Natural Language Processing, and Learning from Noisy Datasets.

References

[1] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. Van Kleef, S. Auer *et al.*, "Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia," *Semantic Web*, vol. 6, no. 2, pp. 167–195, 2015.

[2] D. Vrandečić and M. Krötzsch, "Wikidata: a free collaborative knowledge base," 2014.

[3] F. M. Suchanek, G. Kasneci, and G. Weikum, "Yago: A large ontology from wikipedia and wordnet," *Journal of Web Semantics*, vol. 6, no. 3, pp. 203–217, 2008.

[4] V. Yadav and S. Bethard, "A survey on recent advances in named entity recognition from deep learning models," in *Proceedings of the 27th International Conference on Computational Linguistics*, 2018, pp. 2145–2158.

[5] S. Pawar, G. K. Palshikar, and P. Bhattacharyya, "Relation extraction: A survey," *arXiv preprint arXiv:1712.05191*, 2017.

[6] C. Alt, A. Gabryszak, and L. Hennig, "TACRED revisited: A thorough evaluation of the TACRED relation extraction task," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2020, pp. 1558–1569. [Online]. Available: https://www.aclweb.org/anthology/2020.acl-main.142

[7] Y. Zhang, V. Zhong, D. Chen, G. Angeli, and C. D. Manning, "Position-aware attention and supervised data improve slot filling," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP 2017)*. Association for Computational Linguistics, 2017, pp. 35–45. [Online]. Available: https://nlp.stanford.edu/pubs/zhang2017tacred.pdf

[8] G. Stoica, E. A. Platanios, and B. Poczos, "Re-tacred: Addressing shortcomings of the tacred dataset," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, 2021, pp. 13 843–13 850.

[9] M. Joshi, D. Chen, Y. Liu, D. S. Weld, L. Zettlemoyer, and O. Levy, "Spanbert: Improving pre-training by representing and predicting spans," *Transactions of the Association for Computational Linguistics*, vol. 8, pp. 64–77, 2020.

[10] A. Parekh, A. Anand, and A. Awekar, "Taxonomical hierarchy of canonicalized relations from multiple knowledge bases," in *Proceedings of the 7th ACM IKDD CoDS and 25th COMAD*, 2020, p. 200–203.

[11] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[12] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[13] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the association for computational linguistics*, vol. 5, pp. 135–146, 2017.

[14] J. L. Elman, "Finding structure in time," *Cognitive science*, vol. 14, no. 2, pp. 179–211, 1990.

[15] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[16] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[17] Y. Kim, "Convolutional neural networks for sentence classification," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1746–1751. [Online]. Available: https://aclanthology.org/D14-1181

[18] T. H. Nguyen and R. Grishman, "Relation extraction: Perspective from convolutional neural networks," in *Proceedings of the 1st workshop on vector space modeling for natural language processing*, 2015, pp. 39–48.

[19] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," vol. 30, 2017.

[20] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[21] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[22] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[23] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," 2018.

[24] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the NAACL:HLT 2019, Volume 1 (Long and Short Papers)*, 2019, pp. 4171–4186.

[25] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," 2019.

[26] I. Hendrickx, S. N. Kim, Z. Kozareva, P. Nakov, D. Ó Séaghdha, S. Padó, M. Pennacchiotti, L. Romano, and S. Szpakowicz, "Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals," in *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions*, 2009, pp. 94–99.

[27] A. Mitchell, S. Strassel, S. Huang, and R. Zakhary, "Ace 2004 multilingual training corpus," *Linguistic Data Consortium, Philadelphia*, vol. 1, pp. 1–1, 2005.

[28] M. Mintz, S. Bills, R. Snow, and D. Jurafsky, "Distant supervision for relation extraction without labeled data," in *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*. Association for Computational Linguistics, 2009, pp. 1003–1011.

[29] S. Riedel, L. Yao, and A. McCallum, "Modeling relations and their mentions without labeled text," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2010, pp. 148–163.

[30] R. Yangarber and R. Grishman, "Nyu: Description of the proteus/pet system as used for muc-7 st," in *Seventh Message Understanding Conference (MUC-7): Proceedings of a Conference Held in Fairfax, Virginia, April 29-May 1, 1998*, 1998.

[31] F. Xu, H. Uszkoreit, and H. Li, "A seed-driven bottom-up machine learning framework for extracting relations of various complexity," in *Proceedings of the 45th annual meeting of the Association of Computational Linguistics*, 2007, pp. 584–591.

[32] S. Kim, M. Jeong, and G. G. Lee, "A local tree alignment approach to relation extraction of multiple arguments," *Information processing & management*, vol. 47, no. 4, pp. 593–605, 2011.

[33] G. Zhou, J. Su, J. Zhang, and M. Zhang, "Exploring various knowledge in relation extraction," in *Proceedings of the 43rd annual meeting of the association for computational linguistics (ACL'05)*, 2005, pp. 427–434.

[34] D. P. Nguyen, Y. Matsuo, and M. Ishizuka, "Relation extraction from wikipedia using subtree mining," in *Proceedings of the National Conference on Artificial Intelligence*, vol. 22, 2007, p. 1414.

[35] B. Rink and S. Harabagiu, "Utd: Classifying semantic relations by combining lexical and semantic resources," in *Proceedings of the 5th International Workshop on Semantic Evaluation*. Association for Computational Linguistics, 2010, pp. 256–259.

[36] A. Culotta and J. Sorensen, "Dependency tree kernels for relation extraction," in *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, 2004, pp. 423–429.

[37] R. Bunescu and R. Mooney, "A shortest path dependency kernel for relation extraction," in *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, 2005, pp. 724–731.

[38] R. Mooney and R. Bunescu, "Subsequence kernels for relation extraction," in *Advances in Neural Information Processing Systems*, vol. 18, 2005.

[39] M. Wang, "A re-examination of dependency path kernels for relation extraction," in *Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-II*, 2008.

[40] N. Kambhatla, "Combining lexical, syntactic, and semantic features with maximum entropy models for information extraction," in *Proceedings of the ACL Interactive Poster and Demonstration Sessions*, 2004, pp. 178–181.

[41] R. Socher, B. Huval, C. D. Manning, and A. Y. Ng, "Semantic compositionality through recursive matrix-vector spaces," in *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*, 2012, pp. 1201–1211.

[42] C. Liu, W. Sun, W. Chao, and W. Che, "Convolution neural network for relation extraction," in *International Conference on Advanced Data Mining and Applications*, 2013, pp. 231–242.

[43] D. Zeng, K. Liu, S. Lai, G. Zhou, and J. Zhao, "Relation classification via convolutional deep neural network," in *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. Association for Computational Linguistics, 2014, pp. 2335–2344.

[44] S. Zhang, D. Zheng, X. Hu, and M. Yang, "Bidirectional long short-term memory networks for relation classification," in *Proceedings of the 29th Pacific Asia conference on language, information and computation*, 2015, pp. 73–78.

[45] P. Zhou, W. Shi, J. Tian, Z. Qi, B. Li, H. Hao, and B. Xu, "Attention-based bidirectional long short-term memory networks for relation classification," in *Proceedings of 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016*, 2016, pp. 207–212.

[46] L. Wang, Z. Cao, G. De Melo, and Z. Liu, "Relation classification via multi-level attention cnns," in *Proceedings of the 54th Annual Meeting of the ACL (Volume 1: Long Papers)*, 2016, pp. 1298–1307.

[47] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[48] C. Alt, M. Hübner, and L. Hennig, "Improving relation extraction by pre-trained language representations," in *Automated Knowledge Base Construction (AKBC)*, 2019.

[49] M. E. Peters, M. Neumann, R. Logan, R. Schwartz, V. Joshi, S. Singh, and N. A. Smith, "Knowledge enhanced contextual word representations," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 43–54.

[50] W. Zhou and M. Chen, "An improved baseline for sentence-level relation extraction," *arXiv preprint arXiv:2102.01373*, 2021.

[51] I. Yamada, A. Asai, H. Shindo, H. Takeda, and Y. Matsumoto, "Luke: Deep contextualized entity representations with entity-aware self-attention," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 6442–6454.

[52] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, "Learning entity and relation embeddings for knowledge graph completion," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 29, 2015.

[53] C. Xiong, R. Power, and J. Callan, "Explicit semantic ranking for academic search via knowledge graph embedding," in *Proceedings of the 26th International Conference on World Wide Web*, 2017, pp. 1271–1279.

[54] B. Y. Lin, X. Chen, J. Chen, and X. Ren, "Kagnet: Knowledge-aware graph networks for commonsense reasoning," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 2829–2839.

[55] W. Cui, Y. Xiao, H. Wang, Y. Song, S.-w. Hwang, and W. Wang, "Kbqa: learning question answering over qa corpora and knowledge bases," *arXiv preprint arXiv:1903.02419*, 2019.

[56] X. Han, H. Zhu, P. Yu, Z. Wang, Y. Yao, Z. Liu, and M. Sun, "Fewrel: A large-scale supervised few-shot relation classification dataset with state-of-the-art evaluation," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 4803–4809.

[57] Y. Zhang, P. Qi, and C. D. Manning, "Graph convolution over pruned dependency trees improves relation extraction," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 2205–2215.

[58] L. B. Soares, N. Fitzgerald, J. Ling, and T. Kwiatkowski, "Matching the blanks: Distributional similarity for relation learning," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 2895–2905.

[59] Z. Zhang, X. Han, Z. Liu, X. Jiang, M. Sun, and Q. Liu, "Ernie: Enhanced language representation with informative entities," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 1441–1451.

[60] X. Han, P. Yu, Z. Liu, M. Sun, and P. Li, "Hierarchical relation extraction with coarse-to-fine grained attention," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 2236–2245.

[61] R. Takanobu, T. Zhang, J. Liu, and M. Huang, "A hierarchical framework for relation extraction with reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 7072–7079.

[62] T. Wang, Y. Li, K. Bontcheva, H. Cunningham, and J. Wang, "Automatic extraction of hierarchical relations from text," in *European Semantic Web Conference*, 2006, pp. 215–229.

[63] J. Chen, L. Liu, J. Xu, and B. Hui, "Hierarchical relation extraction based on bidirectional long short-term memory networks," in *Proceedings of the 2019 International Conference on Data Mining and Machine Learning*, 2019, p. 110–113.

[64] P. Jain, P. Kumar, S. Chakrabarti *et al.*, "Type-sensitive knowledge base inference without explicit type supervision," in *Proceedings of the 56th Annual Meeting of the ACL (Volume 2: Short Papers)*, 2018, pp. 75–80.

[65] H. Peng, T. Gao, X. Han, Y. Lin, P. Li, Z. Liu, M. Sun, and J. Zhou, "Learning from context or names? an empirical study on neural relation extraction," in *Proceedings of the EMNLP 2020*, 2020, pp. 3661–3672.

[66] J. Wehrmann, R. Cerri, and R. Barros, "Hierarchical multi-label classification networks," in *International Conference on Machine Learning*. PMLR, 2018, pp. 5075–5084.

[67] D. Mekala, V. Gangal, and J. Shang, "Coarse2fine: Fine-grained text classification on coarsely-grained annotated data," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2021, pp. 583–594.

[68] A. Ghosh, H. Kumar, and P. Sastry, "Robust loss functions under label noise for deep neural networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, 2017.

[69] D. Song, J. Xu, J. Pang, and H. Huang, "Classifier-adaptation knowledge distillation framework for relation extraction and event detection with imbalanced data," *Information Sciences*, vol. 573, pp. 222–238, 2021.

[70] S. Park and H. Kim, "Improving sentence-level relation extraction through curriculum learning," *arXiv e-prints*, pp. arXiv–2107, 2021.

[71] B. Xu, L. Zhang, Z. Mao, Q. Wang, H. Xie, and Y. Zhang, "Curriculum learning for natural language understanding," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 6095–6104.

[72] D. Chen, J. Bolton, and C. D. Manning, "A thorough examination of the cnn/daily mail reading comprehension task," in *Proceedings of 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016*, 2016, pp. 2358–2367.

[73] J. Barnes, L. Øvrelid, and E. Velldal, "Sentiment analysis is not solved! assessing and probing sentiment classification," in *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, 2019, pp. 12–23.

[74] B. Frénay and M. Verleysen, "Classification in the presence of label noise: a survey," *IEEE transactions on neural networks and learning systems*, vol. 25, no. 5, pp. 845–869, 2013.

[75] Z. Guo, Y. Zhang, and W. Lu, "Attention guided graph convolutional networks for relation extraction," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 241–251.

[76] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of machine learning research*, vol. 9, no. 11, 2008.

[77] C. E. Brodley and M. A. Friedl, "Identifying mislabeled training data," *Journal of artificial intelligence research*, vol. 11, pp. 131–167, 1999.

[78] M. Abdar, F. Pourpanah, S. Hussain, D. Rezazadegan, L. Liu, M. Ghavamzadeh, P. Fieguth, X. Cao, A. Khosravi, U. R. Acharya *et al.*, "A review of uncertainty quantification in deep learning: Techniques, applications and challenges," *Information Fusion*, vol. 76, pp. 243–297, 2021.

[79] K. Lee, S. Yun, K. Lee, H. Lee, B. Li, and J. Shin, "Robust inference via generative classifiers for handling noisy labels," in *International Conference on Machine Learning*. PMLR, 2019, pp. 3763–3772.

[80] H. Wei, L. Tao, R. Xie, and B. An, "Open-set label noise can improve robustness against inherent label noise," in *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[81] Q. Wang, B. Han, T. Liu, G. Niu, J. Yang, and C. Gong, "Tackling instance-dependent label noise via a universal probabilistic model," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, 2021, pp. 10 183–10 191.

[82] Y. Bai, E. Yang, B. Han, Y. Yang, J. Li, Y. Mao, G. Niu, and T. Liu, "Understanding and improving early stopping for learning with noisy labels," in *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[83] S. B. Huffman, "Learning information extraction patterns from examples," in *International Joint Conference on Artificial Intelligence*. Springer, 1995, pp. 246–260.

[84] N. Nakashole, G. Weikum, and F. Suchanek, "Patty: A taxonomy of relational patterns with semantic types," in *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 2012, pp. 1135–1145.

[85] M. Miwa and M. Bansal, "End-to-end relation extraction using lstms on sequences and tree structures," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016*, 2016, pp. 1105–1116.

[86] Y. Yao, Z. Sun, C. Zhang, F. Shen, Q. Wu, J. Zhang, and Z. Tang, "Jo-src: A contrastive approach for combating noisy labels," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 5192–5201.

[87] P. Chen, J. Ye, G. Chen, J. Zhao, and P.-A. Heng, "Beyond class-conditional assumption: A primary attempt to combat instance-dependent label noise," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, 2021, pp. 11 442–11 450.