Università degli Studi di Brescia
Dipartimento di Ingegneria dell'Informazione
Dottorato di Ricerca in Ingegneria dell'Informazione
XXXIII Ciclo

Knowledge Transfer Techniques in Deep Learning for Biomedical Named Entity Recognition

Supervisore
Prof. Ivan Serina

Dottorando
Tahir Mehmood
matricola 72195

Relatori
Prof. Alfonso E. Gerevini
Dr. Alberto Lavelli

Settore Scientifico Disciplinare ING-INF/05

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF NOTATIONS/ABBREVIATIONS ALONG WITH FIRST OCCURRENCE

$BioNER$:	Biomedical named entity recognition (pg.xiii)
STM :	Single-task learning (pg.xiii)
MTM :	Multi-task model (pg.xiii)
NLP :	Natural language processing (pg.1)
NER :	Named entity recognition (pg.1)
IE :	Information extraction (pg.1)
DL :	Deep learning (pg.3)
MTL :	Multi-task learning (pg.4)
CRF :	Conditional random field (pg.17)
SVM :	Support vector machine (pg.20)
HMM :	Hidden Markov model (pg.20)
$BiLSTM$:	Bidirectional long short-term memory (pg.22)
CNN :	Convolutional neural network (pg.22)
$ReLU$:	Rectified linear unit (pg.26)
RNN :	Recurrent neural network (pg.27)

$LSTM$:	Long short-term memory (pg.28)
$MTM\text{-}CNN$:	MTM with convolution neural network (pg.39)
$MTM\text{-}CW$:	MTM with character and word inputs (pg.40)
$MTM\text{-}CNN^{ch}$:	MTM-CNN contains word-level and character-level input representations (pg.46)
$MTM\text{-}CNN^{ca}$:	MTM-CNN contains word-level and case-level input representations (pg.46)
$MTM\text{-}CNN^{\bigstar}$:	MTM-CNN trained with GENIA-POS tagging and CoNLL chunking (pg.49)
$MTM\text{-}CNN^{\bigstar^{in}}$:	MTM-CNN trained with GENIA-POS tagging and CoNLL chunking at the innermost layer (pg.49)
$MTM\text{-}CNN^{in}$:	MTM-CNN trained with BioNER task at the innermost layer (pg.50)
$MTM\text{-}CW^{ca}$:	MTM-CW with additional case inputs (pg.56)
$MTM\text{-}CW^{soft}$:	MTM-CW with Softmax function at the output layer (pg.56)
$MTM \rightarrow STM$:	MTM fine-tuned for task specific STM (pg.61)
MTM^{10}:	MTM trained for 10 epochs (pg.63)
$MTM^{10} \rightarrow STM$:	MTM trained for 10 epochs is fine-tuned for task specific STM (pg.63)
MTM^{20}:	MTM trained for 20 epochs (pg.63)
$MTM^{20} \rightarrow STM$:	MTM trained for 20 epochs is fine-tuned for task specific STM (pg.63)

MTM^{cmp}:	MTM trained for complete epochs (pg.63)
$MTM^{cmp} \rightarrow STM$:	MTM trained for complete epochs is fine-tuned for task specific STM (pg.63)
SM :	Student model (pg.73)
MSE :	Mean squared error (pg.74)
$\mathcal{H}$:	Cross-entropy (pg.74)
σ :	Softmax function (pg.74)
SM^{ψ}:	Student model trained with logits of the Softmax-based teacher MTM (pg.75)
SM^{ϕ}:	Student model trained with logits of the ensemble teacher Softmax-based MTMs (pg.78)
MTM^{CRF}:	Teacher MTM with CRF at the output layer (pg.78)
$SM^{\S}$:	Student model trained with logits of the ensemble teacher Softmax-based MTM and CRF-based MTM (pg.78)
$SM^{\dagger\dagger}$:	Student model trained with task specific intermediate BiLSTM layer of Softmax-based MTM (pg.81)
$SM^{\dagger\bigstar}$:	Student model trained with shared and task specific intermediate BiLSTM layer of Softmax-based MTM (pg.82)
$SM^{\bigstar}$:	CRF-based SM trained on the logits of the CRF-based MTM (pg.83)
$SM^{\bigstar\phi}$:	CRF-based SM trained on the logits of the Softmax-based MTM (pg.83)
$SM^{\P}$:	SM trained with soft labels of teacher MTM (pg.86)

τ:	Temperature parameter used to normalized the logits (pg.35)

ABSTRACT

Deep learning methods have produced promising results in many tasks including biomedical named entity recognition (BioNER). Nevertheless, the complex structure of biomedical text data is still a challenging aspect for deep learning models. Additionally, limited annotated biomedical data is available to train these models that have millions of trainable parameters. The Single task model (STM) has difficulties in learning complex feature representations from a limited amount of annotated data. Producing manually annotated data is a time-consuming job instead, there are some efficient ways to train deep learning models on the available annotated data. This work leverages the performance of the BioNER task by using three different knowledge transfer techniques: the multi-task learning method, transfer learning method, and knowledge distillation method. The work presents two multi-task models (MTMs), which learn shared features and task-specific features by implementing the shared and task-specific layers. Jointly training various models helps different tasks to transfer their knowledge implicitly using a shared layer(s).

Due to the implicit feature learning ability of MTM, the MTM is fine-tuned for a specific dataset to tailor general feature representations to a specialized feature representation. The MTM is trained for different epochs as an auxiliary task which is then further fine-tuned for a specific task.

The generalization of MTM is further exploited using knowledge distillation, where MTM supervises the training of a student model. Deep learning models learn generic level features to abstract level features layer by layer. The proposed knowledge distillation approach, therefore, performs knowl-

edge distillation from different layers of MTM that include a shared layer(s) and task-specific layer(s). Additionally, an ensemble method has been implemented where logits of the different MTMs are averaged to train the student model.

The experimental results are analysed using the Friedman's statistical test to evaluate the effect of the experiments in terms of statistical significance. The empirical results and statistical analysis from this work show that these approaches enhance the performance of an STM.

SOMMARIO

I metodi di Deep Learning hanno prodotto risultati promettenti in molti task, incluso il riconoscimento di entità biomediche (Biomedical Named Entity Recognition, o BioNER). Nonostante ciò, la struttura complessa del testo biomedico è ancora un aspetto impegnativo per tali modelli. Oltre a questo, per l'addestramento di questi modelli, che hanno milioni di parametri da allenare, è disponibile solamente una quantità limitata di testi biomedici annotati. I modelli a singolo task (Single Task Model, o STM) hanno difficoltà a imparare caratteristiche complesse da una quantità limitata di dati, la cui produzione è oltretutto molto dispendiosa in termini di tempo. Esistono tuttavia dei modi per allenare modelli di deep learning in caso di scarsa disponibilità di dati. Questa tesi migliora le prestazioni per il task BioNER usando tre diverse techniche di trasferimento della conoscenza: l'apprendimento multi-task, il transfer learning, ovvero l'utilizzo di un modello già addestrato per un altro compito, e la knowledge distillation, ovvero il trasferimento di conoscenza da uno più grande a uno più piccolo. Questa tesi presenta due modelli multi-task (MTM) che imparano caratteristiche sia condivise tra più task che specifiche di uno solo di essi, con l'ausilio di livelli, anche questi condivisi o specifici. L'allenamento contemporaneamente di più modelli aiuta il trasferimento di conoscenza, usando i livelli condivisi.

Grazie all'abilità implicita di imparare caratteristiche dei dati del MTM, successivamente quest'ultimo viene specializzato su un dataset singolo affinchè il modello comprenda le sue caratteristiche specifiche. Il MTM è allenato prima su un compito ausiliario e poi specializzato su un task specifico.

La capacità del MTM di generalizzare è ulteriormente sfruttata con la

tecnica di knowledge distillation, in cui il MTM supervisiona l'addestramento di un "modello studente". Dato che i modelli di deep learning imparano le caratteristiche dei dati livello per livello, l'approccio di knowledge distillation proposto estrae conoscenza dei diversi livelli di MTM, siano essi condivisi o specifici. Inoltre, è stato implementato un metodo di ensemble, in cui i risultati di diversi MTM vengono considerati per l'allenamento del modello studente.

I risultati sperimentali sono stati analizzati usando il test statistico di Friedman per valutare i diversi esperimenti anche in base alla significatività statistica delle differenze tra i loro risultati. I risultati empirici e l'analisi statistica di questa tesi mostrano come questi approcci migliorino le prestazioni dei STM.

1. INTRODUCTION

1.1 Background and Motivation

Online text data carry valuable information and keep expanding very rapidly. However, a significant portion of this data belongs to unstructured forms, and manually dealing with such a large amount of free text is challenging and problematic. As such, intelligent techniques are required to process them according to the problem domain. Natural language processing (NLP), a subfield of artificial intelligence, is used to process the unstructured text data fulfilling the users' needs. NLP enables computers to comprehend, interpret, and manipulate human languages and is being applied to various tasks, including topic discovery and modeling, sentiment analysis, information extraction, among others. Information extraction (IE) is a technique for extracting relevant data from unstructured text. IE comprises various subtasks, one of which is known as named entity recognition (NER). Named entities refer to the proper nouns presented in the sentences. NER recognizes text of interest and labels them into predefined categories such as person, geographical location, organization, etc. NER is considered a sequence labeling problem that determines the output tag of the input words presented in the sentence [2].

The biomedical literature is publishing at an increasing rate, and thus information extraction has become a critical activity in the biomedical domain. Biomedical named entity recognition (BioNER) identifies the medical concepts and categorizes them into predefined categories such as genes, chemicals, diseases, etc., as shown in Figure 1.1

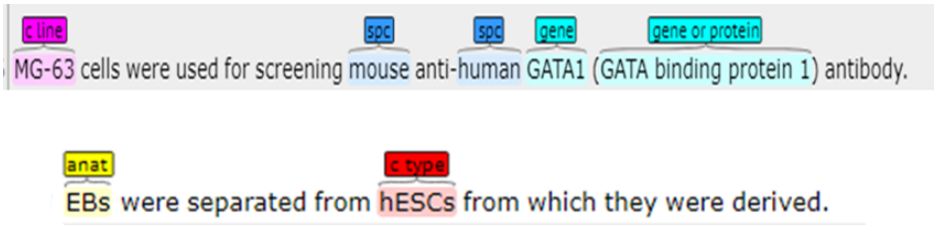


Fig. 1.1: An illustration of Biomedical Named Entity Recognition Task

Biomedical texts are different from the standard text data (e.g. newspaper articles) in several ways, making BioNER more challenging than a standard NER task. Although there are some conventions for writing biomedical concepts, no strict rules exist for the biomedical domain. With open and growing biomedical literature, it becomes more challenging to follow the same naming convention.

First, spelling variations of the entities are found more frequently in the biomedical literature. For example, *N-acetylcysteine, N-acetyl-cysteine, N-acetyl cysteine, acetylcysteine,* and *NAcetylCysteine* refer to the same medicine name with minor variations [3, 4]. Similarly, “IL12", “IL 12", or “IL-12" represent the same entity with different alternative approaches in the text writing [5]. Furthermore, different synonyms refer to the same entity, e.g., *PTEN* and *MMAC1* both represent the same gene entity. Such alternative approaches for representing a single entity are dealt with differently by different independent tokenizers for breaking raw texts into tokens.

Second, there are entities with long compound-word expressions that contain numerous types of characters. For instance, *“10-Ethyl-5-methyl-5,10-dideazaaminopterin"* and *“12-o-tetradecanoylphorbol 13-acetate"* etc., contain alphanumeric and special characters. Different tokenizers produce different outputs for the same entity as they handle these special characters independently. Similarly, the descriptive entities, e.g. *“Pigment epithelium-derived factor"*, *“Medullary thymic epithelial cells"*, etc. make it challenging for entity boundary identification. Nested entities are also found in the

biomedical domain e.g., *"CIITA mRNA"* symbolizes an RNA mention, however, *"CIIT"* refers to DNA [6].

Third, another problematic element of the biomedical literature is acronyms, which often may refer to different entities. For instance, *"TCF"* can refer to *"Tissue Culture Fluid"* or to *"T cell factor"* [7]. Similarly, *"EGFR"* can stand for *"estimated glomerular filtration rate"* or to *"epidermal growth factor receptor"*. Since some occurrences of these entities are dependent on the context of the sentence, training a BioNER system to recognize them becomes more challenging. Furthermore, different human annotators even with the same background can sometimes define the exact synonyms with different medical concepts, e.g., *"p53"* corresponds to a protein in GENIA corpus. In contrast, HUGA nomenclature annotates it as a gene *"TP53"* [8]. Therefore, the capitalization feature of the entities in biomedical literature does not provide valuable information about the entity.

Finally, some of the entities share one head noun along with the disjunction or conjunction construction, e.g. *"91 and 84 kDa proteins"* corresponds to *"91 kDa protein" and "84 kDa protein"* [9].

In view of the limitations above, as mentioned earlier, the BioNER task is more challenging compared to the standard NER task. The early BioNER methods (e.g., dictionary-based and rule-based approaches) are effective but their performance is still limited against open and growing biomedical literature. As compared to the dictionary-based and rule-based methods, the classical machine learning algorithms have shown improved results. The machine learning algorithms require an extensive hand-crafted feature engineering phase that has direct impact on the performance of the models. The performance enhances with more discriminating features, while redundant and irrelevant ones degrade the performance.

The state-of-the-art techniques are based on deep learning methods, which somehow eliminate the need of hand-crafted features while still producing desired results. Deep learning (DL) architectures consist of many layers,

through which these systems learn the complex structure of the data and features layers by layers. The implicit feature learning ability of the DL models has been successful in different fields [10, 11, 12]. Moreover, with the introduction of the distributed representation of the words, also called word embedding, the performance of these systems has been significantly improved in natural language processing.

1.2 Contribution of the Thesis

Deep learning models have shown promising results for BioNER. However, there is still room for improvement due to the peculiar characteristics of the biomedical unstructured text data. Moreover, deep learning models have millions of parameters and their performance commonly gets better when trained on large input data while comparatively only a small quantity of annotated biomedical text data is available. The process of creating human-annotated biomedical text data is both costly and time-consuming.

Hence, this thesis addresses these limitations by employing multi-task learning, transfer learning, and knowledge distillation techniques to provide supplementary knowledge to the learning model during the training. In multi-task learning, different datasets are trained jointly. This increases the amount of training data and also overcomes the overfitting as the multi-task model (MTM) has to generalize on different datasets. In this research, two different multi-task models are implemented that have shown improved performance compared to the single-task model and state-of-the-art MTMs.

The challenges faced in the multi-task learning (MTL) are coped with the transfer learning approach. Transfer learning is a sequential way to transfer knowledge, whereas multi-task learning is a parallel way of sharing knowledge among different tasks. This thesis adopted the transfer learning technique by utilizing the multi-task learning approach. The MTM is trained for different epochs as an auxiliary task which is then further fine-tuned for a specific task.

The purpose is to (first) train the model on general data before tailoring it to a specific dataset. The results obtained in this thesis show that integrating the MTL approach with transfer learning improves performance compared to the MTM and single-task model approaches.

This thesis further utilizes the knowledge distillation technique to leverage the performance of the single-task model (STM). In knowledge distillation, the teacher model teaches the student model through its knowledge representations at different layers. In the proposed knowledge distillation method, an STM is trained on the feature representation of the MTM at different layers. In this way, STM learns through actual labels and knowledge representations distilled from MTM. This knowledge is distilled at different layers of MTM, including shared and task specific layers and Softmax probability distribution.

1.3 Thesis Organization

The chapters of this thesis can be read independently. Chapter 2 gives an overviews of the previous work in biomedical named entity recognition. The chapter also describes the primary deep learning networks that are adopted in the methodology of this thesis. Chapter 3 presents the detailed methodology in which two multi-task learning models are discussed, along with their detailed results. Chapter 4 presents the transfer learning approach, which is the second methodology of this thesis. Chapter 5 discusses the last methodology of the thesis, which is knowledge distillation. While, chapter 6 concludes the thesis and provides recommendations as well as future research directions.

2. RELATED WORK

The biomedical named entity recognition (BioNER) task recognizes the biomedical concepts and categorizes them into predefined categories, e.g. genes, chemicals, diseases, etc. It is crucial to develop datasets that can cover a wide range of biomedical concepts, which can then be used to develop and evaluate BioNER systems. Many of the biomedical datasets were developed for shared tasks targeting the BioNER and the relation extraction task. These datasets are now publicly accepted for evaluating the developed BioNER systems. This chapter introduces those publicly available datasets that are used in the experiments of this thesis. Furthermore, word embedding for a BioNER systems is also discussed in the current part of the research work. The real-valued word vector representation overcomes the “curse of dimensionality" in the classical one-hot word input representation. The current chapter also discusses different approaches adopted for the development of BioNER systems. The earlier approaches for BioNER use methods of dictionary-based, rule-based, and classical machine learning algorithms. The machine learning strategy has shown performance improvement compared to the previous two approaches. The machine learning method favorably leans towards the feature engineering process. The deep learning methods have overcome the feature engineering limitations, and they are now the state-of-the-art techniques for BioNER. This chapter presents an introduction of the selected deep learning methods and of the techniques which will be used later in the proposed methodologies part.

2.1 Statistical Analysis of The Experimental Results

Statistical significance determines the reliability of the results delivered by the experiments whether the obtained results are real or by chance. This statistical significance is computed with p-value that defines the significance level of the achieved results e.g., the obtained results at $p < 0.05$ states that the effect is observed, whereby the p-value greater than 0.05 indicates no effect is observed. In other words, this significance level determines the confidence level for the acquired results. Obtaining results at p-value less than 0.05 means that if the experiments are repeated over and over again then there is a 95% probability that we will get the same results. In this thesis, the Friedman test is performed to determine the statistical significance of the difference among different models' results. The Friedman test was introduced by the American economist Milton Friedman, which is based on ranks concept, that can be used when three or more comparison needs to be performed [13]. As this test is performed on the collection of groups, therefore it is important to precisely underline the pairs of groups that are statistically significant with each other. This can be done using a post hoc (in Latin, it means "after this") analysis where various pairwise comparisons are carried out. In this thesis, the Friedman test is performed to find out the statistical significance of the difference between the results of the models which is followed by a post hoc analysis for multiple pairwise comparisons.

To clarify the application of the Friedman test, an example data[1] is provided in Table 2.1. In this work, the scipy.stats[2] and scikit[3] python packages are used to perform the Friedman test and the post hoc analysis. The post

[1] source:https://wps.prenhall.com/wps/media/objects/11886/12171343/OnlineTopics/bbs12e_onlinetopic_ch12-9.pdf

[2] https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.friedmanchisquare.html#scipy.stats.friedmanchisquare

[3] https://scikit-posthocs.readthedocs.io/en/latest/generated/scikit_posthocs.posthoc_conover_friedman/#scikit_posthocs.posthoc_conover_friedman

	Group 1	Group 2	Group 3	Group 4
Block 1	70	61	82	74
Block 2	77	75	88	76
Block 3	76	67	90	80
Block 4	80	63	96	76
Block 5	84	66	92	84
Block 6	78	68	98	86

Tab. 2.1: An illustration for Friedman test.

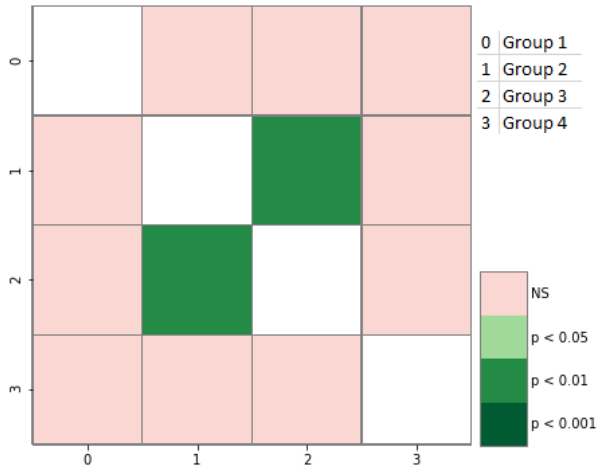


Fig. 2.1: post hoc Conover Friedman test analysis

hoc analysis is only performed if the output of the Friedman test is statistically significant. Figure 2.1 depicts the results of the post hoc analysis on Table 2.1, which is found statistically significant based on the Friedman test. The analysis additionally shows the level of confidence for the pairwise statistically significance which can be 95%, 99%, or 99.9%. The pairwise comparison shows that the difference between Group 2 and Group 3 is statistically significant with $p < 0.001$ (a confidence level of 99.9%) while for others, the pairwise results are not found statistically significant (NS).

To find out which models are statistically better than the others, the ranks of the Friedman test are utilized. The lower rank is assigned to the lowest

	Group 1	Group 2	Group 3	Group 4
Block 1	2	1	4	3
Block 2	3	1	4	2
Block 3	2	1	4	3
Block 4	3	1	4	2
Block 5	2.5	1	4	2.5
Block 6	2	1	4	3
Ranks Total	14.5	6	**24**	*15.5*

Tab. 2.2: Output ranks for Table 2.1

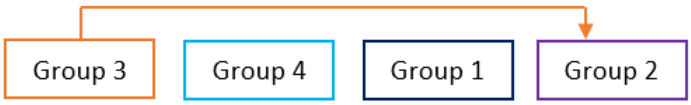


Fig. 2.2: Graphical representation of the Friedman test. The arrows show that the difference between results produced by the different models is statistically significant. Models are shown according to their ranks starting from best model from left to right.

value in each row (the higher the ranks, the better the model) as shown in Table 2.2. In this thesis, such comparison is shown using a graphical representation as shown in Figure 2.2. The models are shown according to their ranks, the leftmost represents the best result while the rightmost is the lowest result. The arrow shows that the difference of the results between Group 2 and Group 3 is statistically significant. While the difference between other results are found not statistically significant. The use of different colors for rectangles (models) and arrows (statistical significance) is just for the readability purpose. The color of the arrow is adapted same as the rectangle that it belongs to.

2.2 Datasets

There are 15 datasets used in the different experiments of the thesis. The bio-entities in these datasets are Chemical, Species, Cell, Gene/Protein, Cell Component, and Disease. Brief descriptions of these datasets are given in Table 2.3. It is important to note that performing human annotation on bio-entities is more difficult than normal text data. The biomedical concepts can be annotated differently depending on the background of the annotators. Considering the annotation of biomedical entities i.e., gene, proteins, and RNA, the human inter-annotator agreement is 70% for these biomedical concepts [14]. This thesis uses the pre-processed form of these datasets[4] where the sentence is represented in the CoNLL[5] column-based format [15]. Each word of the sentence is separated by newline and the first column represents the word token of the sentence, and the second column is the label for such token. The sentences are separated by an empty line. The datasets used in this thesis contain separate training, development, and test sets. The name of the entities and their distribution in the dataset (percentage-wise) are reported in Table 2.4. The values in the table represent the percentage of an individual entity (the O-outside tag is not included) contributing to the train/dev/test file.

2.2.1 AnatEM

The Anatomical Entity Mention (AnatEM) corpus [16], is an extended version of the original Anatomical Entity Mention (AnEM) corpus that also combines Multi-level Event Extraction (MLEE) corpus as well. The AnEM comprises 500 randomly chosen PubMed abstracts and full-text that are annotated for anatomical entity mentions. On the other hand, MLEE comprises 262 PubMed abstracts on cancer's molecular mechanisms linking to angio-

[4] The datasets can be found at the following link `https://github.com/cambridgeltl/MTL-Bioinformatics-2016`

[5] `https://www.clips.uantwerpen.be/conll2003/ner/`

genesis. The AnatEM comprises these two corpora and is further extended by other 100 documents following the AnEM documents selection procedures. Similarly, additional 350 documents were added related to the cancer topics. The selection of these additional documents followed the same process implemented in the MLEE. The final version of AnatEM, therefore, consists in total of 1212 documents.

2.2.2 BC2GM

The BC2GM (BioCreative || Gene Mention) corpus contains a total of 20,000 sentences coming from abstracts of biomedical publications [17]. The BC2GM covers genes, proteins, and other similar entities. However, they are all combined into a single entity class, i.e. Gene.

2.2.3 BC4CHEMD

The BC4CHEMD, BioCreative IV Chemical and Drug, corpus consists of 10,000 abstracts annotated for the single chemical entity containing chemical and drug names [18].

2.2.4 BC5CDR

The BioCreative V Chemical Disease Relation (BC5CDR) [19] comprises of 1500 PubMed articles, of which 1400 articles were selected from 150,000 chemical-disease interactions that were annotated during the Comparative Toxicogenomics Database-Pfizer (CTD-Pfizer) process. The rest of the 100 articles were newly curated and was included in the test set.

2.2.5 BioNLP09

The BioNLP09 is a 2009 shared event task to extract different events among different classes [20]. The named entity was performed via the GENIA event corpus to facilitate the event extraction task. The 10,000 sentences in the

corpus were annotated for protein and related entities into a single entity class, Protein.

Dataset	**Contents**	**Entity counts**
AnatEM	Anatomy NE	13,701
BC2GM	Gene/Protein NE	24,583
BC4CHEMD	Chemical NE	84,310
BC5CDR	Chemical,Disease NEs	Chemical:15,935; Disease:12,852
BioNLP09	Gene/Protein NE	14,963
BioNLP11EPI	Gene/Protein NE	15,811
BioNLP11ID	4 NEs	Gene/Protein:6551; Organism:3471 Chemical:973; Regulon-operon:87
BioNLP13CG	16 NEs	Gene/Protein:7908; Cell:3492; Chemical:2270; Organism:1715; Tissue: 587; Multi-tissue structure:857; Amino acid:135; Cellular component:569; Organism substance: 283; Organ: 421; Pathological formation:228; Immaterial anatomical entity:102; Organism subdivision:98; Anatomical system:41; Cancer:2582; Developing anatomical structure:35
BioNLP13GE	Gene/Protein NE	12,057
BioNLP13PC	4 NEs	Gene/Protein:10,891; Chemical:2487; Complex:1502; Cellular component:1013
CRAFT	6 NEs	SO:18,974; Gene/Protein:16,064; CL:5495; Taxonomy:6868; Chemical:6053; GO-CC:4180
Ex-PTM	Gene/Protein NE	4698
JNLPBA	5 NEs	Gene/Protein:35,336; DNA:10,589; Cell Type:8639l; Cell Line:4330; RNA:1069
Linnaeus	Species NE	4263
NCBI-Disease	Disease NE	6881

Tab. 2.3: Datasets description [1].

Dataset	Entities Name	Train+Dev Set	Test Set
AnatEM	Anatomy	7.241	7.865
BC2GM	Gene	10.505	10.526
BC4CHEMD	Chemical	7.284	7.162
BC5CDR	Chemical	6.061	5.622
	Disease	5.971	5.740
BioNLP09	Protein	9.573	10.274
BioNLP11EPI	Protein	7.662	7.840
BioNLP11ID	Reulon-operon	0.047	0.131
	Chemical	7.036	0.700
	Organism	4.421	3.801
	Protein	4.575	4.134
BioNLP13CG	Gene_or_gene_product	9.975	9.236
	Cancer	2.423	2.896
	Amino_acid	0.088	0.123
	Simple_Chemical	2.631	2.550
	Organism	1.462	1.209
	Cell	4.464	3.987
	Tissue	0.579	0.559
	Organ	0.262	0.328
	Multi_tissue_structure	0.818	0.881
	Cellular_component	0.479	0.472
	Pathological_formation	0.191	0.241
	Immaterial_anatomical	0.075	0.078
	Organism_subdivision	0.060	0.091
	Anatomical_system	0.036	0.049
	Developing_anatomical_structure	0.018	0.040
	Organism_substance	0.197	0.238
BioNLP13GE	Protein	8.100	7.781
BioNLP13PC	Gene_or_gene_product	13.447	13.268
	Simple_chemical	3.272	3.571
	Complex	3.190	3.232
	Cellular_component	0.889	0.879
CRAFT	SO	4.330	3.860
	GGP	4.240	4.320
	Taxon	1.280	1.160
	CHEBI	1.210	1.250
	CL	1.330	1.190
	GO	0.960	0.990
Ex-PTM	Protein	7.967	7.616
JNLPBA	Protein	11.190	9.740
	DNA	5.130	2.810
	Cell_type	3.140	4.860
	Cell_line	2.780	1.470
	RNA	0.504	0.300
Linnaeus	Species	1.153	1.350
NCBI-Disease	Disease	8.220	8.356

Tab. 2.4: Entities percentage distribution in training+development and test dataset

2.2.6 BioNLP 2011 Shared Task

The BioNLP 2009 shared task was extended and presented again in 2011. The BioNLP 2011 shared task covered various tasks including infection diseases (ID), Epigenetics and Post-translational Modifications (EPI), and exhaustive post-translational modifications (Ex-PTM). The BioNLP11EPI events targeted the statements covering modifications in protein and DNA, and their reverse reactions as well, covering 1200 abstracts [21]. Ex-PTM covered more post-traslational modifications in protein related literature, databases, and ontologies total of 360 PubMed abstracts [22]. The BioNLP11ID task enclosed the biomolecular mechanisms of infections that comprises of 30 full articles [23].

2.2.7 BioNLP 2013 Shared Task

The BioNLP 2013 shared task datasets; Cancer Genetics (BioNLP13CG), GENIA Event Extraction (BioNLP13GE), and Pathway Curation (BioNLP13PC) were three tasks out of six tasks in total [24]. The BioNLP13CG task aims to extract the information associated with cancer, e.g., cellular, tissue, etc. The BioNLP13CG contains 600 abstracts from PubMed and is annotated for more than 17,000 events [25], while BioNLP13GE dataset consists of 34 full articles gathered from PubMed Central [26, 27]. The BioNLP13PC dataset annotated for about 16,000 events and contains 525 PubMed abstracts [28, 29] that were collected covering specific pathway reactions based on the pathway models from BioModels and Pathway DB [30].

2.2.8 CRAFT

The Colorado Richly Annotated Full Text (CRAFT) corpus contains 67 full-text articles from PubMed Central Open Access Subset, which were then manually annotated [31]. These articles accumulate over 21,000 sentences, over 560,000 tokens, and approximately 100,000 concept annotations that

contain different biomedical ontologies.

2.2.9 JNLPBA

The JNLPBA corpus was developed for a joint workshop on NLP in Biomedicine and its Applications, which comprised of 2000 abstracts in the train set, while 404 abstracts in the test set that make approximately 22,400 sentences. JNLPBA is developed from the GENIA corpus; however, unlike the GENIA corpus that consist of 36 classes, the JNLPBA only includes 5 classes [32].

2.2.10 Linnaeus

The Linnaeus corpus contains 100 full-text papers, selected randomly from the PMC open access set [33]. The entity mentions presented in the corpus are annotated manually, which are normalized according to the NCBI taxonomy. The corpus contains species mentions, however, 72% of these mentions do not contain direct species information, e.g., patients, child, etc.

2.2.11 NCBI-disease

The NCBI-disease corpus has annotated disease mentions from 793 PubMed abstracts [34]. The corpus consists of 790 unique disease mentions; 698 from MeSH (698) while 92 from OMIM. Furthermore, 91% of the unique concepts are single disease concepts, while the rest contain a combination of concepts.

2.3 Word Vector Representation

In the past, words were represented by a one-hot encoding scheme, where categorical variables are vectorized differently, as shown in Figure 2.3[6]. The one-hot encoding is a sparse vector consisting of 0s and a single 1, which represents the corresponding word. The insertion of a new vocabulary item

[6] source: https://speakerdeck.com/marcobonzanini/word-embeddings-for-natural-language-processing-in-python-at-london-python-meetup?slide=14

Rome = [1, 0, 0, 0, 0, …, 0]

Paris = [0, 1, 0, 0, 0, …, 0]

Italy = [0, 0, 1, 0, 0, …, 0]

France = [0, 0, 0, 1, 0, …, 0]

Fig. 2.3: An illustration of one-hot encoded representation.

in this system requires affixing the new number (0/1) to the vector in order to represent that word. Specifically, the vector size increases with the size of the vocabulary. With the huge vocabulary, the size of the vector also increases drastically, which leads to the "curse of dimensionality". Furthermore, there is no way to represent the relationships between pair(s) of words. Words belonging to specific categories cannot be represented close to each other in one-hot encoding, and 1 is placed at different positions corresponding to the different words. Such representation does not provide any information, and therefore, the dot product of any different words yields an output of 0, which is not a valuable information.

These limitations are overcome by representing a word in the dense vector. This dense representation, also called distributed word embedding representation or simple word embedding, has the ability to store the semantic meaning of the word in a real-valued vector. Thus, the word embedding distributes the properties of the word in a low-dimensional real-valued vector, as shown in Figure 2.4. Such representation gives more beneficial information about the word. For instance, in Figure 2.4, it can be noticed that the beginning embedding values of capital names have approximately the same values showing that they represent the same category. A similar representation can be found for country embedding vectors as well. Additionally, the last value of the embedding vector represents the connection between these different words. The real-valued embedding vector contains useful information, and performing some mathematical operations on these vectors yield more

Rome = [0.91, 0.83, 0.17, …, 0.41]

Paris = [0.92, 0.82, 0.17, …, 0.98]

Italy = [0.32, 0.77, 0.67, …, 0.42]

France = [0.33, 0.78, 0.66, …, 0.97]

Fig. 2.4: An illustration of word embedding representation.

valuable information, e.g., Paris+Italy−France ≈ Rome[7]. Furthermore, the dimension of the word embedding vector is smaller (usually ranging from 50 to 300) than the size of the vocabulary.

Usually, word embedding is learned by feeding corpus to a feed-forward neural network that learns the words' similarity through back-propagation. The neural network outputs the weight of the first layer as an embedding of a fixed dimension. Nowadays, there are deep learning APIs that offer embedding layers for deep learning models, which also learn the word embedding during training. However, these kinds of embedding have limited usage mostly for the specific experiments. However, there are some available neural network models that explicitly learn the embedding of the words trained on massive corpora [35, 36].

2.4 Conditional Random Field

Conditional random fields (CRFs) belong to the discriminative models in which conditional probability $p(y/x)$ is computed for the given X [37]. For every X_i corresponding Y_i is predicted, where X is the evidence variable while Y is called the label variable. CRF considers the neighboring states or contextual information and therefore is best suited for the prediction tasks.

[7] source: https://speakerdeck.com/marcobonzanini/word-embeddings-for-natural-language-processing-in-python-at-london-python-meetup?slide=14

This may help the learning model to generalize well for the prediction task. For this reason, in the label sequencing problems, such as NER and PoS tagging, many of the researchers have preferred CRF over Softmax. For any input, the Softmax function calculates the output probability distribution for a specific single input (e.g., single word) ignoring the neighboring inputs information. As such the Softmax function may omits the useful information lying within other words.

2.5 BioNER Methods

2.5.1 Dictionary-based Approaches

In a dictionary-based approach, the entities present in the text are extracted using pre-defined entities gazetteer. The dictionary approach, therefore, works as a look-up table. Since this approach is simple and effective, it gives high precision as the text is matched against the entity presented in the dictionary. However, this approach fails when it comes to matching synonyms, homonyms (e.g., abbreviations share lexical forms with common English words), spelling variations (e.g., word order and punctuation), short name entities, and limited coverage for new entities. Hirschman et al. [38] utilized pattern matching technique for gene names recognition using FlyBase. They achieved a recall of 31% for abstracts, while the recall for full articles was reported as 84%. They further explained the main reason for the poor recall that was homonymy, as many lexical forms of the gene names were also common English words (e.g., by, for, an, and can). Tsuruoka and Tsujii [39] addressed the spelling variant issue in the dictionary approach. The authors developed a spelling variant generator that used a probabilistic technique for insertion, deletion, and substitution of digits and characters. These operations basically considered the edit distance for the variant of entities. The entities with less or equal edit distance were categorized as a spelling variant.

2.5.2 Rule-based Approaches

In this approach, predefined rules are considered for the recognition of entities. The rules generally comprise naming structures, i.e., utilizing morpho-syntactic features (e.g., capitalization, word alphanumerical composition, special nouns of special verbs, the presence of special symbol), or through using any lexical or orthographic clues. For this reason, the rule-based approach requires a good understanding of linguistics. Development of the rules changes from domain to domain and it is also a time-consuming job. Hou and Chen [40] filtered out false candidates in order to benefit the gene/protein entity recognition. The filtration process is done by exploiting the gene/protein collocates, which were extracted from biological corpora. Furthermore, to improve the performance, authors combined the results of other available named entity recognizers. The results showed performance improvement for the combined approach where filtering techniques along with integrated named entity systems were used.

Narayanaswamy et al. [41] performed chemical and gene/protein entities recognition. The suffixes and chemical roots were used for entity recognition along with other different terms as features. Furthermore, surrounding and context were also utilized in the entity recognition task. The evaluation was made on manually annotated Medline for 55 abstracts indicating comparable results.

Thomas et al. [42] performed protein named entity recognition. They utilized a cascade of finite-state transducers to perform recognition of even complex entities in numerous stages. Though, the reported performance was not satisfactory, the authors claimed that the adaption of their systems to a new domain could be cost-effective, reliable, and fast.

Ananiadou [43] utilized computational morphological grammar and lexicon for medical term recognition. The authors found that most of the medical terminology involves Latin and Greek words and affixex and suggested four-level ordered morphology for extracting the entity patterns.

2.5.3 Machine Learning Approaches

The Machine learning-based approaches have great advantage for their flexible domain adaption and, at the same time, have depicted promising results over previously mentioned approaches. The machine learning algorithms use features to learn the patterns presented in the data. For this reason, these approaches highly depend on the available features. Therefore, many of the machine learning approaches are still suffering from the generalization issue.

Collier et al. [44] proposed a bi-gram Hidden Markov model (HMM) that utilized character and lexical orthographic features for entity recognition of different classes. The sequence of words presented in the sentence, along with other features, were used as input to the model. Afterward, the probability distribution was calculated for each word and its corresponding classes. The classes with the highest probability for the sequence of words were selected as the final class label for the word. The evaluation of the 100 Medline abstracts showed an F1-score of 73%.

Kazama et al. [45] used features like HMM states, affixes, and Part-of-Speech tags to train their multi-class support vector machine (SVM), model. This experiment was done on the GENIA corpus resulting F1-score of 50%.

Shen et al. [46] also exploited the suffixes, Part-of-Speech (POS) tags, and noun heads features for their HMM-based model. They found performance gain by using POS tags.

Lee et al. [47] proposed a two-phased entity recognition system using a support vector machine. This methodology consists of the boundary extraction and classification of the entities. For multi-class classification, they adopted hierarchical classification using ontology. The word, POS, orthographical characteristics, suffixes were used as features. For boundary identification, the F-score was reported as 74.8%, while for the classification, the reported F-score was 66.7%.

Zhou and Su [48] proposed a BioNER model based on HMM and SVM using the JNLBA dataset. They proposed different features to leverage the

performance of their base model. The use of in-domain POS feature increased the F-score of their model from 60.3 to 64.1. Similarly, they extracted cascade entity features using a rule-based method, which raised the F1-score to 63.4. When they combined all the features, i.e., in domain POS, cascade information, entity dictionaries, name alias resolution, and abbreviation detection resulted in a best F1-score of 72.5.

Finkel et al. [49] used BioNER system based on maximum entropy Markov model. Their proposed system used word's local features along with some external knowledge including domain-specific gazetteers, parsing and searching the web. During training, the label's boundaries of the same entities were merged so that the system can be trained on more examples for each single class. Their systems achieved F1-score of 70% on JNLPBA dataset.

2.5.4 Deep Learning

Machine learning algorithms have shown performance improvement compared to the rule-based and dictionary-based approaches. However, the performance of the machine learning algorithms highly relies on the input representation, which is given in the form of features. The performance enhances with more discriminative features, while redundant and irrelevant features can cause performance degradation. Machine learning algorithms learn the hidden pattern in the data. The development of discriminated input features for machine learning algorithms requires more time and effort. Deep learning models, however, usually do not require feature engineering and perform feature extraction implicitly. Deep learning techniques consist of many layers through which they learn the complex structure of the data and learn the features layers by layers. This advantage has determined the success of deep learning algorithms and therefore, they have been adopted in many different fields. With this great success of deep learning in different fields, many natural language processing researchers have also adopted deep learning models and have obtained promising results.

Habibi et al. [50] proposed a deep learning model that uses long short-term memory to perform BioNER. The model used word and char information in the sentences. The embedding information of both inputs was used to train the model. Furthermore, char embedding was passed through a bidirectional long short-term memory (BiLSTM) to capture implicit features of the characters. Three different pre-trained word embeddings were used for the word embedding; PubMed-PMC, Wiki-PubMed-PMC, and the patent one. The PubMed-PMC embedding was trained on the PubMed abstracts and PMC articles, whereas wiki-PubMed-PMC was trained additionally along with English Wikipedia articles. The wiki-PubMed-PMC and PubMed-PMC were 200-dimensional (200d) embedding. The third embedding was trained using the Gensim embedding toolkit on the coarsely 20,000 European patents having biomedical topics for a 50-dimensional embedding vector. The inputs were further processed with a bi-directional long short-term memory to capture the context of the sentence in both directions. The proposed model used a conditional random field (CRF) at the output layer. The experiments were conducted on 33 different datasets that consist of five different categories of biomedical entities, i.e., Cell lines, Chemicals, Disease, Gene/protein, and Species. The results of the proposed model were compared with the publicly available BioNER and the generic CRF model utilizing some NER features. The proposed model showed performance gain with the wiki-PubMed-PMC word embedding, and therefore authors used the same embedding for their further experiments. The proposed model showed a performance gain up to 5% compared to the generic CRF as well as other available BioNER tools.

Zhu et al. [51] used Convolutional Neural Network (CNN) for their proposed model called GRAM-CNN. Their model extracts numerous features at different levels using different kernel sizes in CNN for the same input. The CNN helped the model to extract the n-gram features for the input words, and which is further processed by the filter to represent the word into a single vector. They used word, character, and part-of-speech tags as

input for their model. A pre-trained word embedding was used for the word embedding, whereas character and part-of-speech tags embeddings were randomly initialized and trained during model training. Moreover, to extract the character-level features, a CNN was used on top of the character embedding. Three different datasets, BC2GM, NCBI, and JNLPBA, containing numerous categories, were used in the experiments. The results for the datasets were mixed. For BC2GM and JNLPBA, on the one hand, the proposed model did not show performance improvement against the ensemble machine learning approach, but it outperformed some of the (non)ensemble ones and some deep learning approaches. On the other hand, their approach showed performance gain for the NCBI dataset.

Yoon et al. [52] proposed a model called CollaboNet that comprises of numerous BiLSTM-CRF. The idea was to develop a model that can perform BioNER for multiple entities and can identify polysemous entities. The different single BiLSTM-CRF model was trained for each dataset and then combined during CollaboNet training. The proposed model then exchanges the inference of other models to help the target-specific model to decide which is the best entity type for the specific word. If the inference model correctly labels the input, then the target model does not label that word. The word and char information were used to train the model. Furthermore, CNN was further used to capture the orthographic features of the characters; hence the model is the same used in the [50], where BiLSTM was used to process char level information. Five different datasets were used during the experiments consisting of the Chemicals, Disease, Gene/Protein categories. The proposed approach showed performance improvement compared to some state-of-the-art models. However, the author trained a separate model for each category, performing BioNER on a single class. Training on a single category instead of a whole dataset reduces the ambiguity presented in the dataset due to different categories. This is one of the main reasons that most of the BioNER fails to achieve better results on a dataset with different categories.

Crichton et al. [1] proposed a multi-task learning approach for BioNER. The proposed model used a CNN based deep learning model. The word embedding information is also processed by the convolution layer, followed by the fully connected layer, whereas the Softmax layer was used for output labeling. The model for the single-task learning approach and multi-task learning approach was the same. For the multi-task learning approach, two flavors were utilized; in the first approach, the model shared the upper layers of the model among all the datasets while fully connected layers and Softmax layers were kept private. The second approach induced the information of the auxiliary task where a single-task model is trained on part-of-speech tags and used its fully connected layer output to train the multi-task and other datasets. They took advantage of the multi-task learning approach to train on 15 different biomedical datasets. The results showed a substantial performance gain over the single-task model. However, comparing with the benchmark results, their results were not very prominent. The possible reasons could be using only word information for the training of the model. When the model only considers the word embedding information for training, the model can trap in the out-of-vocabulary problem where the specific word is not found in the static word embedding. The general approach to overcome this issue is to use the character embedding information.

Wang et al. [53] also adopted the multi-task learning approach and used the same 15 datasets used by Chrichton et al. [1]. The basic model is the same model used by the [50] and also resembled the model proposed in [52] except that Wang et al. used BiLSTM to process character embedding information, whereas in [52] a CNN was used to process the character embedding information. The authors adopted different approaches for training the multi-task model. In the first approach, they shared only the character-level information among all the tasks, while in the second approach, they shared the word-level information and BiLSTM, whereas, in the last method, they shared both the character-level information and the word-level information along with the

BiLSTM among all the tasks. They found that the last approach showed performance gain against the first two approaches and compared it to the other benchmark methods, including the results of Chrichton et al. [1].

Another multi-task approach is adopted by Wang et al. [54] where the base model is same as presented in [52], utilizing a Character-CNN, BiLSTM, and CRF. They utilized a different variation in their multi-task learning approach. This involves using shared layers as well as introducing private layers at the same level, introducing calculations of different losses at different levels and also using gated interaction unit (performs element-wise multiplication and sigmoid function). All these MTM models were trained on a pair of two datasets. The experiments were performed on the datasets implemented in [1]. However, they filtered the datasets into main and auxiliary parts. The primary dataset contains only a single category of the entity, whereas the auxiliary dataset contains a different category of entities. For this reason, their proposed MTMs show some performance gain as the dataset contains a single category of entity which does not carry much ambiguity compared to the dataset having instances of different categories. However, using more than two datasets in their approach caused performance degradation for many datasets.

Giorgi and Badar [55] took advantage of the transfer learning where they partially trained their deep learning model on the silver standard corpora and then further fine-tuned on the gold standard corpora. The baseline model is the same as presented in [50]. They collected abstracts from CALBC-SSC-III-Small corpus targeting gene/protein, chemicals/drugs, living beings, and diseases. They trained the model on source data with the parameters that avoid the model from generalizing on the source data. The model was then generalized with different parameters on the 23 gold standard corpora. These datasets contain entities belonging to the Genes/proteins, Species, Diseases, and Chemicals. They noticed a notable increase, with a fine-tuned method, in the F1-score, compared to the same model trained solely on the target

datasets. They find out that transfer learning was more advantageous to those target datasets, with few labels.

2.6 Deep Neural Networks and Techniques

2.6.1 Convolutional Neural Network

Artificial neural networks have been effectively used in different nonlinear problems. With the availability of huge data and more computational power, neural networks with more complex structure are proposed called, deep neural networks. Since then, it has been used in different domain and become state-of-the-art technique. Different deep neural networks are proposed and convolutional neural network (CNN) is one them. CNN is a massive breakthrough in deep learning. It has proven a most effective neural network structure in image recognition and classification [10]. CNN is developed for both extracting features and classification as the dual-task that extracts and learns features layers by layers. The initial layers learn the low-level features, whereas complex features are learned as learning process moves towards the depth of the layers. The CNN basically performs four operations: convolution, activation, pooling, and fully connected layers [56]. The convolution operation involves application of the filter on the pixels. This is the dot product of the pixel with the filter weight. Feature mapping for the original input image is generated for each filter as filter slide through the data. The output of the convolution operation is passed through the activation function. The activation function applied on the convolution output is the Rectified Linear Unit (ReLU), which is a non-linear function. The activation step is followed by the pooling (subsampling or downsampling); an operation that reduces the dimensionality of each feature map trying to keep the salient features. The final operation consists of the output classification which is performed by the fully connected layers. Despite that the CNN was mainly developed for computer vision, it is also used in the natural language processing as well.

However, instead of pixels, the CNN utilizes the text information, usually in the form of words or characters, and extracts the hidden features. The matrix of words represents the pixels information just like in computer vision. However, the NLP problem is different from the aspect of computer vision, where neighbouring pixels could be semantically associated and sharing the identical object. In NLP the orders of the words could modify the meaning of the sentence e.g., adjective changing the noun. Usually the CNN is used to capture N-gram information of the text. The CNN builds the feature maps by moving filters across text representation e.g., word embedding or character embedding.

2.6.2 Recurrent Neural Network and Long Short-Term Memory

The CNN has shown some encouraging results in NLP; however, there are many limitations using CNN in some of the NLP applications. The CNN cannot be used in the dynamic length of the sentences. Furthermore, the neurons in the same layer of the CNN are not connected to each other; thus, they cannot communicate with each other and only passes the information to the next layer. In a problem like named entity recognition, it is important to consider the neighboring words for output labeling of the words. This cannot be achieved in the CNN, where neurons are not connected at the same layer. The recurrent neural network (RNN) overcomes this limitation. The RNN works similarly as we humans do. We listen to the sentence from the first word till it's end, to drive the meaning of the sentence.

The RNN also considers the previous word's information and the current processing word, thus sharing the features learned at different occurrences of the word. Figure 2.5 represents the structure of the RNN, where x and y are the inputs and outputs, respectively. The u, v, and w represent the weights whereas R is the hidden neural unit of the RNN. The hidden unit calculates the hidden state, $h_t = f(Ux_t + Wh_{t-1})$, at time stamp t. It considers the current input x_t and previous hidden state, h_{t-1}, whereas f represents a

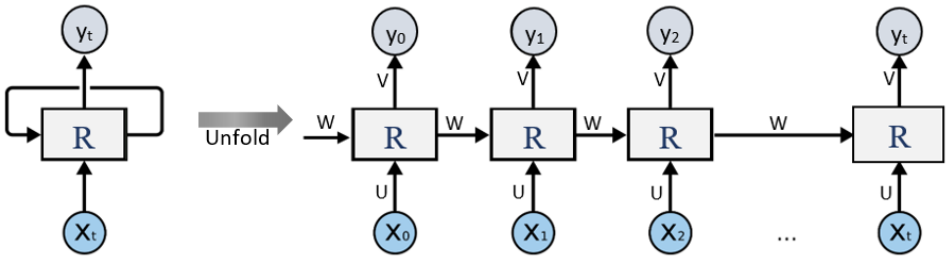


Fig. 2.5: Recurrent neural network and unfolded version during execution.

function that can be linear or nonlinear e.g., *tanh* or *ReLU*. While the output for time stamp t is calculated as $y_t = Vh_t$.

This helps the RNN to remember the previous word information in the text. In other words, the RNN uses memory to retain the previous information. This enables the RNN to correlate the relationship between different events that occurred at different time stamps. However, the simple RNN suffers from word dependency, when it comes to the very long sentences where the last words have a relationship with the beginning ones. Furthermore, the information in RNN passes sequentially to the next neuron. Propagating information from many neurons and performing the operation on each step makes that information shadier. In other words, the RNN embraces the gradient vanishing problem.

To overcome this limitation, long short-term memory (LSTM), a variant of RNN, was first introduced by Hochreiter and Schmidhuber [57]. The LSTM is typically developed to overcome the long dependency problem and retains the information for a long time. This is accomplished using a separate memory cell called, cell state, which sequentially passes through all LSTM units. Furthermore, the LSTM performs more operations, using multiple neural networks, compared to the simple RNN hidden state.

In general, the LSTM uses three gates to process the information, as shown in Figure 2.6. The input gate concatenates the previous hidden state with the current inputs and decides which value of the input that should be

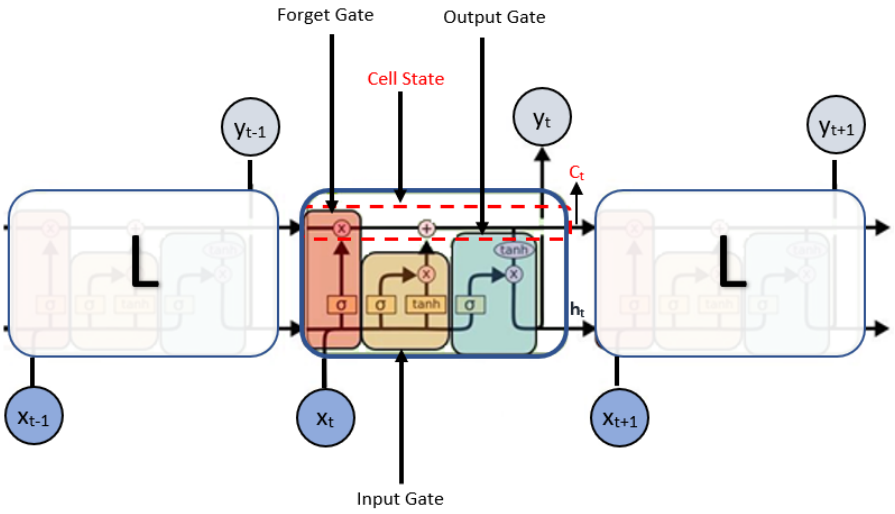


Fig. 2.6: Structure of long short-term memory Network

used and by how much via using the Sigmoid function. Furthermore, the weight is also considered for the input which is determined by *tanh* function. The forget cell decides which information should be forgotten. It grasps the previous cell output and current input which is concatenated with the previous hidden state. This information is process using a Sigmoid function that decides which information needs to be forgotten. The outputs of both; the forget gate and input gate are combined to construct the new current state. Finally, the output gate performs the Sigmoid function on the current input and previous hidden, multiplying with *tanh* output of the current cell state. The calculations can be summarized in the following equations:

$$Forget\ Gate\ f_t = \sigma(W_f.[h_{t-1}, x_t] + b_f)$$

$$Output\ Gate\ o_t = \sigma(W_0.[h_{t-1}, x_t] + b_0) \quad \text{and } h_t = o_t * tanh(C_t)$$

$$Input\ Gate\ i_t = \sigma(W_i.[h_{t-1}, x_t] + b_i)$$

$$New\ Temporary\ Cell\ State\ \tilde{C}_t = tanh(W_c.[h_{t-1}, x_t] + b_c)$$

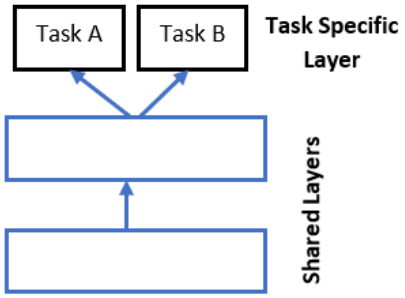
$$Current\ Cell\ State\ C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

The LSTM usually learns from the beginning of the sentence until it's end in a forward manner. However, in purpose to get the full context of the sentence, it is better to look through it in a backward manner too. This is done using bidirectional LSTM (BiLSTM), where two hidden units are used to get the information of the sentence in a forward and backward way.

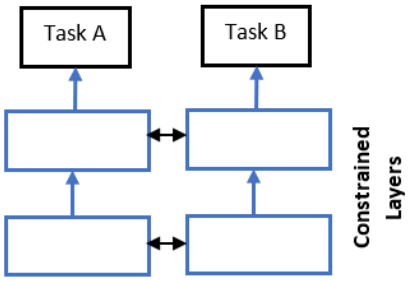
2.6.3 Multi-task Learning

In general, the deep learning model performance highly depends on the amount of annotated data. The model performs better when a large amount of data is available. Unfortunately, in different biomedical tasks only a limited quantity of annotated text data is available and in this case deep learning models are unable to generalize well. Producing manually annotated data is expensive and time-consuming job. One solution to such limitation is to take advantage of other related tasks that share common features. In a single-task learning approach, related tasks cannot get benefits from the other task. By using a single-task model we cannot share the training signal with the other related-task.

Multi-task learning (MTL) is an approach where different tasks share their knowledge among themselves, thus help to leverage the performance of another task. MTL is an inductive learning process that learns the generalization by utilizing the knowledge of different tasks [58]. When tasks are sufficiently related, they can provide an inductive bias that forces models to learn generally useful representations. In MTL related tasks provide inductive bias which guides model to discover more common representations [59]. Two different methods are used in the MTL approach i.e., hard parameter sharing and soft parameter sharing, and is shown in Figure 2.7. Hard parameter sharing is the most common method used in MTL where complete sharing (i.e., parameters) of hidden layers among different tasks is done. In soft parameter sharing, separate models are created for different tasks. These models are then somehow enforced to loosely match parameters of the shared



(a) Hard Parameters shared MTM



(b) Soft Parameters shared MTM

Fig. 2.7: Hard parameters vs Soft parameters shared MTMs

layers most commonly done by regularizing the parameters of the shared layers.

The MTM can be seen as an implicit data augmentation technique as well. Jointly training of various models help them to transfer their knowledge implicitly using shared layer(s). The MTL strategy, therefore, increases the size of the data available to the MTM. The MTL also allows model to learn those features which can be more challenging to learn independently from any specific task. In other words, the approach of task B to learn rigorous features can be more complex compared with task A, therefore, the model learns those features from task A. Training jointly on related tasks helps the multi-task model to learn common features among different tasks by using shared layers [60]. The MTL optimizes the model under construction concurrently that allows the model to generalize well for the related tasks. In a single-task learning, a model is more prone to overfitting for a specific task, whereas MTL lowers the chances of overfitting as the model has to learn the common representation for all tasks. Training more tasks bring more generalization for MTM. In MTL, model focuses on more relevant features as some tasks give information about irrelevant and relevant features in a high dimensional and noisy data. Similarly, the noise presented in the dataset has less impact in a MTL approach as all the noises are averaged during

the training. The MTL also guides the model to narrow-down the shared representation supported by many tasks which might be efficient when the purpose is to use the model to learn novel tasks if they are from the same environment [61]. In the MTL approach, some of the layers in the model are shared among various tasks while keeping some layers task-specific. The task-specific layers learn features that are more related to the current task. Training related tasks together help the model to optimize the parameter's value.

2.6.4 Transfer Learning

Transfer learning involves transferring the knowledge from one domain to another [62]. Usually, the model is trained on a task in one domain which is then re-used on another related domain or related task [11]. The MTM can also be seen as transfer learning but in MTM the tasks are learned simultaneously. In contrast, in the transfer learning tasks are learned sequentially. The transfer learning is done in two stages: pretraining stage and domain adaptation stage. The pretraining stage involves training of the base model which is then reused on the target task in the adaptation phase. The pretraining phase, although, is expensive but is usually required to perform once. Therefore, it is best practice to choose the source task that can exhibit general representations for many target tasks.

In transfer learning, the model is trained on an auxiliary task which is then re-used on the main task. Similarly, the model can be trained on a source domain which can then be re-used on the target domain. For instance, the model can be trained on the book reviews and then re-used on hotel reviews, in this case source and target domains are different but source and target tasks are the same. Similarly, the source and domains can be same while the source and target tasks are different e.g., the object detection model can be used for image classification. In a third case, both domains as well as both tasks are different e.g., spam classifier is used for radiology text report

classification.

In transfer learning, the pre-trained model can be used as feature extractor or model weight's initialization. The feature extractor works similar to the feature engineer process, however, this process is done by a deep learning model instead of performing manually as seen in the general machine learning approaches. In a fixed feature extraction mechanism, the output of the pre-trained layers are used in the main task. In this case, the weights of the pre-trained layers are kept frozen and usually layers before the final output layer are used. This can be perceive as input features to the new model. The feature extraction is useful when the tasks used in the transfer learning are similar to each other. The feature extraction method is required when the purpose is to learn the features once for all tasks and, therefore, saves the time as it is not necessary to compute the features again for the new tasks. These features are mostly the low-level features e.g., dots or lines in an image.

The transfer learning could involve a fine-tuning method, where weights of the pre-trained layers are used in the main task model and then the whole model is fine-tuned. In this case, the weights of the layers are not kept frozen. The idea is to re-learn new features rather than learning from scratch. The naive example could be to learn the numbers after five instead of learning numbers again from one. The fine-tuning method actually fine-tunes the general-purpose representation to task specific representation. This method is convenient when the purpose is to implement the pre-trained model for many different tasks.

Yosinski et al. [11] performed experiments to compare the feature extractor and the fine-tuning techniques. They found that for the feature extractor, the performance of the main task model depends on where the layers are cut. Researchers concluded that keeping top layers' weight frozen can be helpful for similar tasks. The frozen weights of the middle layers show performance degradation because of the complex co-adaptations they learn. At last, keep-

ing the weights of the lower layers do not show much performance degradation as these layers are more general. While investigating the fine-tuning method, they found it useful without any strict constraint of layers cut at any level of the base model.

2.6.5 Knowledge Distillation

In transfer learning, the learned representation from the source domain is utilized in another related domain. In contrast, the objective of knowledge distillation is to train a model with the knowledge learned by another model. The idea of knowledge distillation is to train a simple (student) model on the knowledge learned by the complex (teacher) model. More specifically, the knowledge distillation approach addresses how to transfer the generalization of one model, usually a complex model (teacher), to another model, usually a simple model (student). The complex models or ensemble approaches usually produce better results than the simple single-task model, but it is computationally expensive to train them. The knowledge distillation approach helps the simple model (student) to produce better results than the stand alone single model and the ensemble models. In this way the student model can be trained on fewer training examples since it will also consume the knowledge learned by the teacher model during training. The idea is that the complex model has already been generalized on the data during its training. This helps the student model to achieve or nearly achieve the generalization of the teacher model. The student model not only learns through the gradient of itself but also though the gradient of another knowledge.

Transferring knowledge from a teacher model is usually done in the shape of the probabilities predicted by the teacher model. The objective of any learning model is to predict the correct class for the input example and assign a high probability to that class whereas allocating small probability values to the rest of the classes. Associating the probabilities to the rest of the incorrect classes is not performed randomly. These side probabilities

also carry information which depicts how a specific model has generalized the classes presented in the dataset. For instance, there is very little chance of miss-classifying a motorbike image into a car image but the probability would still be higher for miss-classifying it into the truck image. The Softmax activation function outputs the probability distribution of the possible classes for the specific instance. The sum of these Softmax probability distributions sums to 1.

These Softmax probabilities give more information compared to the one-hot “hard labels". For instance, the Softmax probabilities, [0.7, 0.2, 0.1], show ranking of the classes. Such information cannot be examined in the hard labels e.g, [1, 0, 0] where we cannot extract any such information. The posterior probabilities can pass an extra useful signal to the student model during its training. However, training the student model to match these probabilities could not be so much useful as the student model can only pay more attention to the highest probability value. To overcome this barrier, it is better to soften these final Softmax output probabilities through normalizing them [12]. The normalized probabilities represent soft labels which provide some knowledge distillation to the student model [63]. The student model then pays attention to other values as well along with the highest probable class. Hinton et al. [12] proposed a tunable-parameter term temperature, τ, to soften the posterior probabilities as given in the Equation 2.1.

Introducing this new, τ, parameter normalizes the output probabilities e.g., setting the value of $\tau = 3$, for the above Softmax output will now yield [0.375, 0.317, 0.307]. It can be notice that, though, the output probabilities are become more soften but the ranking of the potential classes are still unchanged. Furthermore, the student model will pay attention to other values apart from the highest probable class. The large value of τ more softens the Softmax output and enhances the non-target class output probability [64]. Keeping $\tau = 1$ makes it standard Softmax function. The large value of τ more softens the Softmax output and enhances the non-target class out-

put probability [64]. On the downside, it also reduces the probability value of the target class. Therefore, it is vital to choose the right value for the temperature parameter.

$$\text{Softmax}(z_i) = \frac{\exp(z_i/\tau)}{\sum_j \exp(z_j/\tau)} \tag{2.1}$$

2.7 Ensemble Methods

An ensemble method is a method that applies several models and then combines their output to yield a single result. Ensemble approaches, therefore, generally produce more accurate results than a single model [65]. Neural networks can learn nonlinear relationships present in the data. However, this nonlinear behavior is sensitive to the noise present in the data and the initial random weights of the model layers. For this reason, these models produce a different prediction every time they are trained as they produce a different final set of weights each time. This causes a high variance in the predictions. In order to reduce that variance, numerous neural network models are trained which are then combined in an ensemble approach to generate a single prediction. This also overcomes the likely absence of distinctive features representation in any single model as various models can learn different sets of features, and therefore reduces the the generalization error [66].

The final prediction can be performed using majority voting, average, or weighted voting/average schemes. In the majority voting scheme, a final prediction represents the most voted prediction for a specific class by various models whereas, in the averaging method, all models contribute equally in the final prediction. The weighted vote/average is the extension of the previous schemes where the output from different models is weighted individually and combined for the final prediction.

The ensemble approaches have been utilized by many researchers for BioNER. Zhou et al. [67] combined three classifiers consisting of two discrim-

inative Hidden Markov models and a single support vector machine. Their ensemble approach used simple majority voting strategy and showed performance improvement for gene/protein BioNER in BioCreAtIvE task 1*A*. Torii et al. [68] proposed BioTagger-GM based on an ensemble approach where outputs from different systems are combined and a voted scheme is used for the final gene/protein names recognition from literature. Their system resulted best score in the BioCreAtIvE || challenge. Doan et al. [69] aggregated recognition from three different classifiers i.e., support vector machines, conditional random fields, and a rule-based approach and used different voting schemes for the final entity recognition. The proposed system used clinical text and showed that the ensemble approach achieves better results compared to the single classifier.

3. BIONER USING MULTI-TASK LEARNING

The traditional approaches for BioNER use classical machine learning methods like Conditional Random Fields (CRFs), Support Vector Machine (SVM), and Hidden Markov Models (HMMs) [70]. These methods have shown promising results; however, they are highly dependent on handcrafted features [71]. Additionally, adopting machine learning techniques to a new domain would be difficult and requires extra effort to perform domain-specific feature engineering.

To tackle these limitations, deep learning methods have been deployed in the sequence labelling problem. They represent the state-of-the-art techniques used in the BioNER systems, which helped to get rid of the manual feature engineering step and at the same time are able to produce promising results simultaneously. Nevertheless, such methods are still facing challenges due to the complex structure of the biomedical text data. Training deep learning models also need large amounts of input data. In contrast, a comparatively small quantity of annotated biomedical text data is available to train these systems with millions of parameters. Manually annotating biomedical text data is an expensive and time-consuming job. One solution to such a limitation is to take advantage of other related tasks that share common features. Such knowledge sharing can be accomplished with techniques such as multi-task learning and transfer learning.

Multi-task learning (MTL) is an approach where different related tasks are trained simultaneously. In the MTL approach, some of the layers in the model are shared among various tasks while keeping some layers task-specific. Training jointly on related tasks helps the multi-task model (MTM) to learn

common features among different tasks by using shared layers [60]. This also allows the model to generalize well for the related tasks while the task-specific layers learn more associated features to the current task. Training related tasks together helps MTM to optimize the parameters value. In this manner, MTL optimizes the MTM under construction concurrently. Moreover, the MTL lowers the chances of overfitting as the MTM has to learn the common representation among all tasks.

The MTL has been widely applied in many different domains e.g., computer vision [72], speech recognition [73], and drug discovery [74]. Collobert and Weston [75] used CNN-based MTM and trained multiple NLP tasks jointly such as POS tagging, NER, chunking etc. Bollmann and Søgaard [76] showed that, using the MTL approach, their model increased the performance for historical spelling normalization. Peng and Dredze [77] used the MTL approach for different domains, i.e., Chinese word segmentation and named entity recognition. Plank et al. [78] used an auxiliary loss function for rare words and the primary loss function for the POS tagging task, targeting 22 languages including Finnish, French, and English. Yang et al. [79] used the MTL approach to perform different tasks simultaneously, including POS tagging, chunking, and NER in English, Dutch, and Spanish. Zhang and Weiss [80] used POS tagging as a regularizer of input representation for dependency parsing. Johansson [81] performed parsing of multiple treebanks in a shared features representation approach and used one treebank as input to another treebank. Søgaard and Goldberg [82] demonstrated that auxiliary tasks should be used at innermost layers so that the main task can effectively learn from a shared representation. Hashimoto et al. [83] used a hierarchical model to learn different NLP tasks at successively deeper layers jointly.

In this chapter, two different MTMs are proposed to improve the performance of the BioNER. The first proposed model, the multi-task model with convolutional neural network (MTM-CNN), uses two stacked BiLSTMs where one BiLSTM is shared among various tasks, while the second BiLSTM

is task-specific. The MTM-CNN utilizes the word, character, and orthographic features of the input sentences. It is found that the MTL approach is not always effective, and therefore, this research applies different techniques to improve MTM-CNN. Results show that MTM-CNN learns better when trained with additional dissimilar auxiliary (e.g., POS tagging, Chunking) tasks other than the same (BioNER) auxiliary tasks. The second proposed model, the multi-task model with character and word input (MTM-CW), uses the character and word input representation of the input sentences and tries to overcome the catastrophic interference [84]. In MTM-CW, the input representation propagates along with the middle layers information to the subsequent layers. This chapter also evaluates both proposed models using the Friedman statistical test [85], which assigns ranks to the models according to their outputs.

3.1 MTM-CNN

In this section, the multi-task model (MTM-CNN) is proposed that consists of a Convolutional neural network (CNN) layer and BiLSTM layers as shown in Figure 3.1. The proposed MTM-CNN model differs from the model presented by Wang et al. [53] in three ways. First, in the proposed approach an orthographic-level representation of a word is used. Many studies have exploited word's orthographic-level information for their models [86, 87, 88]. The orthographic-level information of the word provides some explicit information to the model, which can enhance the model performance where deep learning models implicitly learn orthographic-level features. This can also help the conditional random field (CRF) whose output highly depends on hand-crafted features [89]. In MTM-CNN, the orthographic-level information is used, speculating that it helps MTM-CNN to extract more information about the entities. In this chapter, the orthographic-level feature is referred to as *case-level features* and both terms can be used interchangeably.

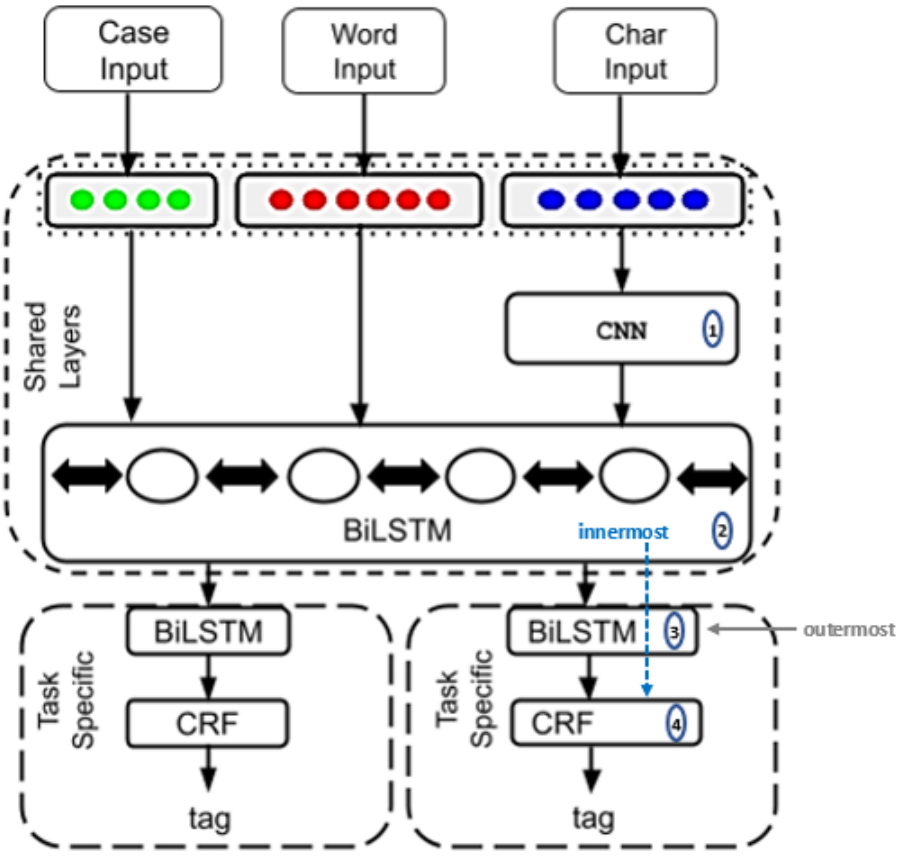


Fig. 3.1: The proposed MTM-CNN (circles represents embeddings). Innermost depicts the task is trained without task-specific BiLSTM. Outermost represents the task is trained with task-specific BiLSTM.

The orthographic-level (case-level) representation in MTM-CNN considers the structure-level information of the word. The case-level features considered in the experiments include the capitalization features of the word e.g., whether all letters in the specific word are capital or small, or if the specific word starts with a capital or small letter, whether the word contains digits or all alphabetic letters etc.

Subsequently, the proposed MTM-CNN utilizes CNN (represented by circled 1 in Figure 3.1) instead of BiLSTM, differently to the Wang et al. [53], to

extract character-level features. Many of the state-of-the-art approaches use CNN at character-level [90, 91] due to its unique feature extraction ability. CNN learns global-level features from local-level ones. This enables CNN to extract more hidden features.

Third, MTM-CNN implements stacked layers of BiLSTMs. Using stacked BiLSTMs helps hidden states of BiLSTM to learn the hidden structure of the data presented at different time stamps. This helps BiLSTM to learn features at more abstract level. The first innermost BiLSTM layer (marked by circled 2 in Figure 3.1) is shared among all the tasks while the second layer of BiLSTM (shown by circled 3 in Figure 3.1) is task-specific. The MTM-CNN implements CRF (represented by circled 4 in the figure) at the output layer for final sequence labeling. The CRF performs tagging of the current token by considering neighboring tags at a sentence level [88]. Yang et al. [92] performed experiments comparing CRF with Softmax and found that CRF produces better results compared to the Softmax. The experiments in this chapter also show that the performance of the MTM-CNN dropped when Softmax is used instead of CRF at the output layer.

Furthermore, two dissimilar auxiliary tasks (other than BioNER) are also introduced that are: GENIA-POS tagging and CoNLL chunking, to investigate their impact on MTM-CNN. The auxiliary tasks are trained in a same manner as other BioNER task i.e., with a task-specific BiLSTM layer (outermost). In the MTL approach, different tasks provide a supervision signal to other tasks. It is important to inspect the proper supervision that can be at any level. Following this hypothesis, the auxiliary tasks are trained at innermost (shared) BiLSTM layer without task-specific BiLSTM. Assuming that training an auxiliary task at the innermost layer makes the shared BiLSTM a complete features representation of that task, which may propagate more useful signals to the task-specific BiLSTM. The same hypothesis is applied on the auxiliary BioNER tasks where they are trained without task-specific BiLSTM.

During the MTM-CNN training, each task is defined with its optimizer, and therefore, the loss function related to the specific task is optimized. It means that the shared layers' parameters and the task-specific ones are changed during the training for the particular task. Optimizing shared layers' parameters for all tasks allows the model to find standard features among different tasks.

3.2 Experimental Settings

The experiments are performed on the 15 datasets mentioned in **section 2.2,** which are also used by Crichton et al. [1] and Wang et al. [53] for their MTMs. The experimental configuration is the same as adopted by the Wang et al.[1] and [93, 94], which use both train and development datasets for training the model.

Moreover, to represent words, a domain-specific pre-trained word embedding is used since the general one can cause a high rate of out-of-vocabulary words. In particular, the WikiPubMed-PMC word embedding is utilized. The WikiPubMed-PMC word embedding is trained on a large set of the PubMedCentral(PMC) articles, PubMed abstracts and English Wikipedia article [55]. Whereas character embedding is initialized randomly and the case embedding is represented by the identity matrix, where each diagonal 1 represents a word's orthographic feature.

The maximum number of epochs is set to 50 with an early stop set to 6. Moreover, each experiment is run 10 times, if not specified otherwise, and the average F1-score is reported in this chapter.

[1] `https://github.com/yuzhimanhua/Multi-BioNER`

Datasets	STM	MTM-CNN
AnatEM	85.89	**86.99**
BC2GM	**80.90**	80.82
BC4CHEMD	**88.60**	87.39
BC5CDR	85.66	**87.85**
BioNLP09	87.03	**88.74**
BioNLP11EPI	81.48	**84.75**
BioNLP11ID	83.21	**87.65**
BioNLP13CG	81.27	**84.25**
BioNLP13GE	73.36	**79.82**
BioNLP13PC	86.33	**88.84**
CRAFT	**83.84**	83.15
Ex-PTM	72.70	**80.95**
JNLPBA	**74.48**	74.05
linnaeus	87.38	**87.79**
NCBI-disease	84.11	**85.66**
Average	82.42	**84.58**

Tab. 3.1: STM vs MTM-CNN

3.3 Results and Discussions of MTM-CNN

As a first step, a single-task model (STM) is implemented for all 15 datasets mentioned in **section 2.2**. Afterward, MTM-CNN is trained with all 15 datasets in a MTL approach. The best results are shown in **boldfont** while second best is represented by *Italic* style. Table 3.1 depicts the comparison between the results of MTM-CNN and its counterpart STM. Each experiment is run 10 times, and the average F1-score of those 10 runs is reported in this chapter.

It can be seen that for most of the datasets, the results are improved markedly by using MTM-CNN, showing the importance of the MTL ap-

Datasets	Wang et al.	Crichton et al.	STM
AnatEM	*85.30*	81.55	**85.89**
BC2GM	*80.00*	72.63	**80.90**
BC4CHEMD	**88.75**	82.95	*88.60*
BC5CDR	**86.96**	83.66	*85.66*
BioNLP09	*84.22*	83.90	**87.03**
BioNLP11EPI	77.67	77.72	**81.48**
BioNLP11ID	74.60	*81.50*	**83.21**
BioNLP13CG	**81.84**	76.74	*81.27*
BioNLP13GE	69.30	*73.28*	**73.36**
BioNLP13PC	*85.46*	80.61	**86.33**
CRAFT	*81.20*	79.55	**83.84**
Ex-PTM	67.66	*68.56*	**72.70**
JNLPBA	*72.17*	69.60	**74.48**
linnaeus	*86.94*	83.98	**87.38**
NCBI-disease	*83.92*	80.26	**84.11**
Average	80.40	78.43	**82.42**

Tab. 3.2: Single task model results comparison

proach in BioNER. The BC2GM, BC4CHEMD, CRAFT, and JNLPBA show performance degradation with the MTL approach. One possible reason could be the size of these datases. The size of these datasets is big compared to the rest of the other datasets. For this reason, a performance increase is noticed for those datasets that have a small number of entity annotations. This can be seen for Ex-PTM which has a small number of entities and shows a noticeable improvement with the MTL approach. The results suggest that MTM-CNN leverage the performance of those datasets, which do not have many examples. The results illustrate that the MTM-CNN can learn complex features that are difficult to learn in a STM.

Furthermore, Table 3.2 shows the comparison between the results of dif-

ferent state-of-the-art STMs. It can be seen that, on most datasets, STM yields better performance compared to others while the model proposed by Wang et al. [53] performed well on four of these datasets. The proposed model by Crichton et al. [1] is unable to show improvement on any dataset. The model proposed by Crichton is CNN-based and does not consider the character-level information and may have resulted out-of-vocabulary error. This might be the reason that their model is unable to show performance gain compared to the results of Wang et al. and MTM-CNN.

Table 3.3 represents the comparison of different MTMs. It can be seen that for all the datasets, the MTM-CNN model outperforms the one proposed by Crichton et al. [1] with a notable difference of F1-score up to 4%. Whereas compared to the multi-task model presented by Wang et al. [53], MTM-CNN attained better results for most of the datasets, while for the rest, the results are comparable.

3.3.1 Effects of Different Inputs Representations of a Sentence

As previously described, Wang et al. [53] do not use case-level information while MTM-CNN includes it. To see the impact of the case-level information, the MTM-CNN is run with different input representations. To differentiate between different representations, MTM-CNN is modified with various labels i.e., MTM-CNN, MTM-CNN^{ch}, and MTM-CNN^{ca}. The MTM-CNN contains word-level, character-level, and case-level representations and is the originally proposed model. The MTM-CNN^{ch} contains word-level and char-level input representations, whereas MTM-CNN^{ca} contains word-level and case-level input representations. Table 3.4 reports the results of input representations, where MTM-CNN and MTM-CNN^{ch} have outperformed MTM-CNN^{ca} , while MTM-CNN^{ca} shows improvements only for BC2GM, CRAFT, and Ex-PTM. It is also noted that using only case-level information and word-level information (MTM-CNN^{ca}) shows a performance gain for few datasets compared with the MTM-CNN and MTM-CNN^{ch}. It is interesting to note

Datasets	Wang et al.	Crichton et al.	MTM-CNN
AnatEM	*86.04*	82.21	**86.99**
BC2GM	78.86	73.17	**80.82**
BC4CHEMD	**88.83**	83.02	*87.39*
BC5CDR	**88.14**	83.90	*87.85*
BioNLP09	*88.08*	84.2	**88.74**
BioNLP11EPI	*83.18*	78.86	**84.75**
BioNLP11ID	*83.26*	81.73	**87.65**
BioNLP13CG	*82.48*	78.90	**84.25**
BioNLP13GE	**79.87**	78.58	*79.82*
BioNLP13PC	*88.46*	81.92	**88.84**
CRAFT	*82.89*	79.56	**83.15**
Ex-PTM	*80.19*	74.90	**80.95**
JNLPBA	*72.21*	70.09	**74.05**
linnaeus	**88.88**	84.04	*87.79*
NCBI-disease	*85.54*	80.37	**85.66**
Average	83.79	79.70	**84.58**

Tab. 3.3: Results comparison for different multi-task models

that MTM-CNN^{ch} shows an increase in the F1-score for ten datasets compared to the MTM-CNN^{ca}. However, when the case-level information is embedded into MTM-CNN^{ch} (the model becomes MTM-CNN), the performance degradation is noted for few datasets. Abstractly, including only word and char level information in the proposed architecture can also show better results. However, simply using only case-level information and word-level information causes performance degradation as excluding character-level information causes out-of-vocabulary problem.

3.3.2 Effects of Different Sequence Labeling Functions

We have also modified MTM-CNN using Softmax instead of CRF to determine the impact of Softmax and CRF in the proposed model. Table 3.5 shows

	MTM-CNN	MTM-CNN^{ch}	MTM-CNN^{ca}
Datasets	(word, char, case)	(word, char)	(word, case)
AnatEM	**86.99**	*86.86*	86.43
BC2GM	80.82	*81.00*	**81.01**
BC4CHEMD	*87.39*	**87.66**	87.21
BC5CDR	87.85	**88.08**	*87.93*
BioNLP09	**88.74**	*88.67*	88.64
BioNLP11EPI	*84.75*	**85.17**	84.66
BioNLP11ID	**87.65**	*87.28*	87.01
BioNLP13CG	*84.25*	**84.39**	84.09
BioNLP13GE	79.82	**80.44**	*80.42*
BioNLP13PC	*88.84*	**89.02**	88.59
CRAFT	*83.15*	83.12	**83.94**
Ex-PTM	*80.95*	80.83	**80.97**
JNLPBA	**74.05**	73.93	*73.99*
linnaeus	*87.79*	87.45	**87.88**
NCBI-disease	**85.66**	*85.38*	85.07
Average	84.58	**84.62**	84.52

Tab. 3.4: Results comparison for all MTM-CNN models

the results comparison for Softmax and CRF. It can be seen that MTM-CNN^{Soft} only shows performance gain for the linnaeus dataset compared with the CRF-based MTM (MTM-CNN). The performance degradation of the Softmax function could be due to the ambiguous entities present in the data. The Softmax function produces the output probability distribution for any specific word and ignores the information of the neighbour words. In contrast, CRF considers the whole sequence for output labelling considering neighbouring words information.

Datasets	*MTM-CNN*	*MTM-CNN^{Soft}*
AnatEM	**86.99**	85.84
BC2GM	**80.82**	78.71
BC4CHEMD	**87.39**	84.40
BC5CDR	**87.85**	86.78
BioNLP09	**88.74**	88.00
BioNLP11EPI	**84.75**	83.29
BioNLP11ID	**87.65**	87.34
BioNLP13CG	**84.25**	82.98
BioNLP13GE	**79.82**	79.69
BioNLP13PC	**88.84**	87.79
CRAFT	**83.15**	80.98
Ex-PTM	**80.95**	79.60
JNLPBA	**74.05**	71.52
linnaeus	87.79	**88.37**
NCBI-disease	**85.66**	84.17
Average	**84.58**	83.30

Tab. 3.5: Results comparison of proposed MTM-CNN with CRF and Softmax(MTM-CNN^{soft}) at the output Layer.

3.3.3 Effects of Different Auxiliary Tasks

In the previous experiments, all the auxiliary tasks are the same i.e., performing BioNER task for another dataset during MTM-CNN training. In order to see the effect of different tasks in the MTL approach, the experiments are extended with various tasks and the same BioNER task but at a different level of layers in MTM-CNN.

In this regard, three different approaches are adopted; in the first approach (MTM-CNN$^{\bigstar}$), during the training of MTM-CNN two additional but different auxiliary tasks are introduced: that are GENIA-POS tagging and CoNLL chunking. In a second approach, MTM-CNN$^{\bigstar^{in}}$, these auxiliary

Datasets	*MTM-CNN*	*MTM-CNN*$^{\bigstar}$	*MTM-CNN*$^{\bigstar in}$	*MTM-CNN*^{in}
AnatEM	86.99	*87.14*	**87.35**	86.69
BC2GM	80.82	**81.48**	*81.37*	81.26
BC4CHEMD	87.39	**88.64**	*88.47*	87.95
BC5CDR	87.85	*88.10*	**88.35**	88.01
BioNLP09	88.74	*88.78*	88.76	**88.92**
BioNLP11EPI	*84.75*	84.65	**84.94**	84.53
BioNLP11ID	*87.65*	**88.04**	87.61	87.52
BioNLP13CG	84.25	84.47	*84.59*	**84.61**
BioNLP13GE	79.82	79.41	**80.09**	*80.01*
BioNLP13PC	*88.84*	88.78	**89.06**	88.74
Ex-PTM	**83.15**	81.49	*81.57*	81.13
CRAFT	80.95	*83.66*	**84.17**	83.50
JNLPBA	**74.05**	72.44	*72.63*	72.48
linnaeus	87.79	**88.97**	88.36	*88.49*
NCBI-disease	85.66	85.72	**86.01**	*85.77*
Average	84.58	84.78	**84.89**	84.64

Tab. 3.6: Results comparison of proposed multi-task learning approach with different auxiliary tasks. *MTM-CNN*$^{\bigstar}$ is trained along with GENIA-POS and CoNLL Chunking. *MTM-CNN*$^{\bigstar in}$ the GENIA-POS and CoNLL Chunking auxiliary tasks are trained in the innermost layer. *MTM-CNN*^{in} the BioNER auxiliary tasks trained in the innermost layer.

tasks, GENIA-POS tagging and CoNLL chunking, are trained at the innermost layer. This eliminates the task-specific BiLSTM (layer denoted by 3 in Figure 3.1) for these two auxiliary tasks. Using auxiliary tasks at the innermost layer helps the outermost layer (circled 3) to learn from a complete representation of the auxiliary tasks. The second approach illustrates the performance improvement compared to the first approach, which motivated the third approach of auxiliary tasks of this section.

In the third approach (MTM-CNN^{in}), the auxiliary tasks are the same

BioNER task used in the simple proposed MTM-CNN but this time the auxiliary tasks are trained at the innermost layer (without task-specific BiLSTM) and the main task is used at the outermost layer (having shared BiLSTM layer). If MTM-CNN^{in} is trained for AnatEM then the rest of the datasets (BC2GM, BC4CHEMD etc.) are treated as auxiliary tasks for AnatEM, and do not have task-specific BiLSTM. More specifically, in both the second and third methods, the auxiliary tasks do not have the task-specific BiLSTM layer, while the main tasks have the task-specific BiLSTM layer.

Table 3.6 shows the results of the different approaches using auxiliary tasks. It can be seen that introducing GENIA-POS and CoNLL chunking auxiliary tasks, the results (MTM-CNN$^{\bigstar}$) are improved for ten datasets against MTM-CNN. However, when these tasks are used at the innermost layer (do not have task-specific BiLSTM), it is noticed that the results of the model, MTM-CNN$^{\bigstar in}$, are improved for twelve datasets compared to the proposed MTM-CNN. Similarly, when the same BioNER auxiliary tasks are used at the innermost layer (last column) as in the proposed MTM-CNN^{in}, the results are improved for nine datasets. Conclusively, the MTM-CNN is found more effective when trained with auxiliary tasks other than the same BioNER tasks.

3.3.4 Statistical Analysis of MTM-CNN

To statistically evaluate the performance of the proposed MTM-CNN, the Friedman test is applied to the different models' outputs. The Friedman test is used when three or more comparisons are drawn [13, 85]. The Friedman test ranks the values in the column and uses these rank values to find the significance of the data. Figure 3.2 shows that the results produced by the proposed models and their variants are statistically significant. All MTMs results are statistically better than STM at the confidence level of 99.9%. However, MTM-CNN^{soft} is the only MTM which is not able to produce statistically significant result against STM. It can also be seen that the different approaches

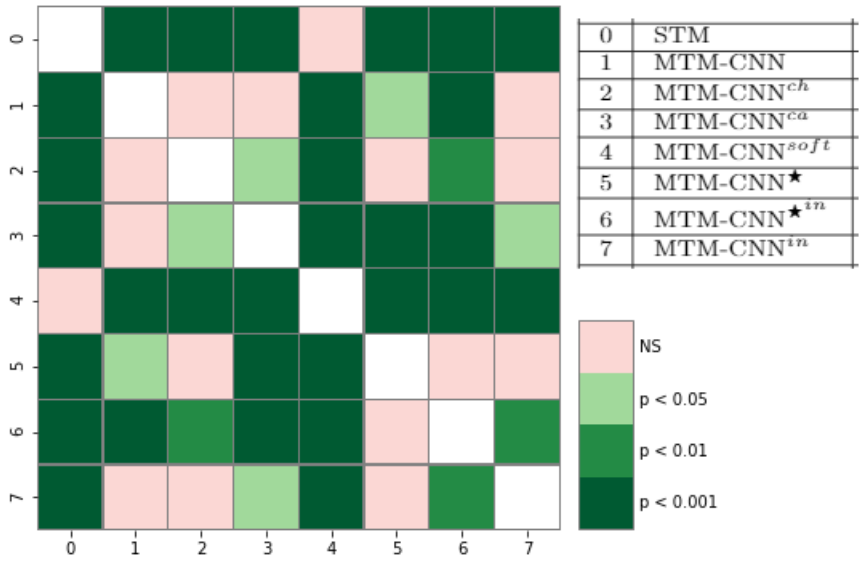


Fig. 3.2: post hoc pairwise analysis with Nemenyi of Friedman test for MTM-CNN.

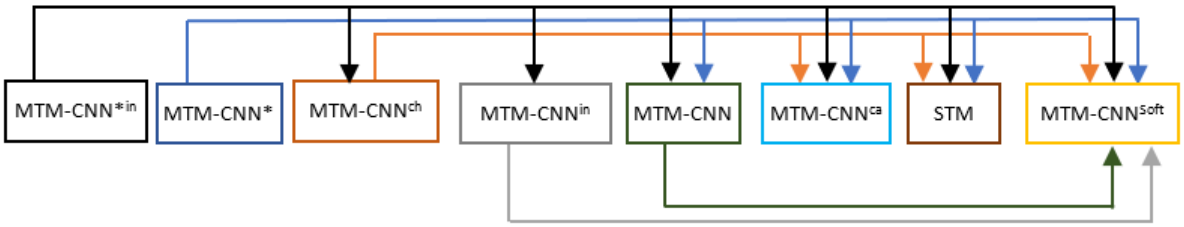


Fig. 3.3: Graphical representation of the Friedman test for MTM-CNN. The arrows show models that are statistically significant with each other. Models are shown according to their ranks starting best model from left to right.

of input representation MTM-CNN,MTM-CNN^{ch}, MTM-CNN^{ca} do not produce a statistically significant result with each other. All MTMs resulted in statistically significant results w.r.t Softmax-based MTM (MTM-CNN^{soft}) with confidence level of 99.9%. The result of MTM-CNN$^{\star in}$ (GENIA-POS and CoNLL chunking used at the innermost layer) is found statistically significant w.r.t to all approaches except for $MTM\text{-}CNN^{\star}$ (GENIA-POS and CoNLL chunking used at the outermost layer).

The output ranks of the Friedman test are considered to analyse which models are statistically superior to other model(s). Figure 3.3 shows the models according to their statistical ranks where the left most represents the best model which decreases from left to right. The arrows represent the statistical significance between the different models. For ease of understanding the figure, different colors are used for the rectangles and arrows. For a specific rectangle (model), the arrows have the same color of the rectangle. It can be seen that the proposed MTM-CNN$^{\bigstar^{in}}$ is statistically better than the rest of the approaches. Using the auxiliary tasks at the innermost layer is found most effective producing statistically significant results w.r.t most of the other approaches. It is also noticed that the proposed method, MTM-CNN^{ch}, with only word and character input representation of the sentence is better than the word and case input representation (MTM-CNN^{ca}).

3.4 MTM-CW

The MTM-CNN [95] presented in section 3.1 comprises the stacked layers of BiLSTM. However, moving towards a deep LSTMs network can cause the gradient vanishing problem as well [96]. Furthermore, using a very deep architecture for some of the tasks could also lead to the catastrophic interference. In the catastrophic interference, the neural network starts forgetting what it has learned. To tackle these issues, a new model MTM-CW is proposed in this section. The proposed multi-task model with character and word input representations (MTM-CW) propagates input embedding information along with the outputs of different shared layers to the subsequent layers as shown in Figure 3.4. This helps successive layers to learn the complex structure from inputs embeddings and encoded representation of the previous layers to overcome the gradient vanishing problem and the catastrophic interference in stacked LSTMs. The skip connections (circled 5 and 6) are represented with dashed arrows in Figure 3.4 and these skip connections make this model

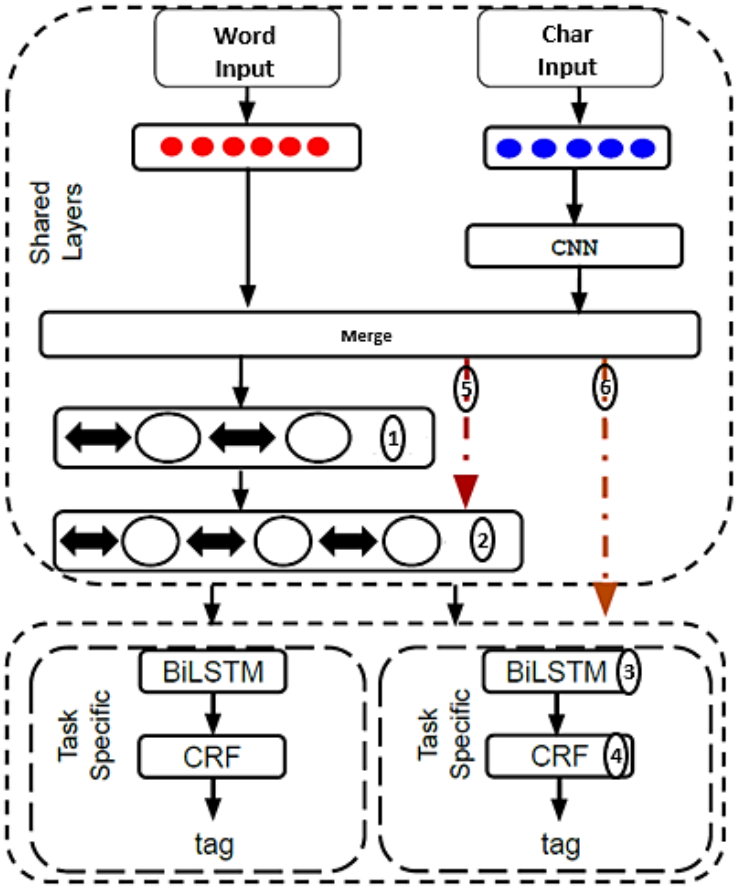


Fig. 3.4: Proposed MTM-CW Model where dashed arrows show skip connections. Circles represents embedding.

different from our previous proposed model. Additionally, the results of section 3.3.1 demonstrated that case information does not improve the results significantly. For this reason, MTM-CW does not use case-level information of the word.

3.5 Results and Discussions of MTM-CW

In Table 3.7, we compare the results produced by MTM-CW with previous approaches [53, 95]. A substantial improvement can be observed in the F1-score for MTM-CW compared to these models. The MTM-CW elevates the F1-score for twelve and eleven datasets compared with [53] and MTM-CNN, respectively. However, to observe whether skip connections (connections from previous layers) have truly contributed to the performance of the model, the skip connections (layers numbered with 5 and 6) are dropped (refer to Figure 3.4). MTM-CW without skip connections (MTM-CW$^{w/o}$) makes it similar to MTM-CNN (section 3.1) but with two shared BiLSTMs. The effect of such variation is reported in Table 3.8, where it can be observed that few datasets show moderate performance increase, while for most datasets the performance drops. This supports the intuition of proposing the MTM-CW, where by propagating the information to the lower layers using skip connections positively impacts the model. Moreover, it is interesting that, even after dropping those skip connections, the MTM-CW$^{w/o}$ is still able to perform better compared to state-of-the-art models. This implies that, with the increasing size of training examples, more layers of LSTM should be considered [96]. For this reason, the proposed model has shown performance improvement compared to [53].

3.5.1 Effects of Different Input Representation and Sequence Labeling Functions

The experiments are extended by introducing the case-level information of a word in MTM-CW. In this work, the case-level feature includes the same information of the word that is considered for the MTM-CNN section 3.3.1 i.e., the capitalization features of the word, whether the word is only numeric, or whether the word contains digits, etc.

A variant of MTM-CW is also proposed where CRF is replaced with

Datasets	Wang et al. [53]	MTM-CNN [95]	MTM-CW
AnatEM	86.04	*86.99*	**87.50**
BC2GM	78.86	*80.82*	**81.57**
BC4CHEMD	*88.83*	87.39	**89.24**
BC5CDR	*88.14*	87.85	**88.54**
BioNLP09	88.08	**88.74**	*88.52*
BioNLP11EPI	83.18	*84.75*	**85.36**
BioNLP11ID	83.26	**87.65**	*87.19*
BioNLP13CG	82.48	*84.25*	**84.94**
BioNLP13GE	*79.87*	79.82	**80.91**
BioNLP13PC	88.46	*88.84*	**89.16**
CRAFT	82.89	*83.15*	**85.23**
Ex-PTM	80.19	*80.95*	**81.72**
JNLPBA	*72.21*	**74.05**	72.10
linnaeus	**88.88**	87.79	*88.12*
NCBI-disease	*85.54*	**85.66**	85.07
Average	83.79	*84.58*	**85.01**

Tab. 3.7: Multi-task models comparison where CW represents character and word respectively.

Softmax at the output layer to understand the impact of both methods on predicting the output label of the entities. Table 3.8 depicts the comparison of both approaches, MTM-CW with case-level information (MTM-CW^{ca}) and MTM-CW with Softmax (MTM-CW^{soft}). The increase in performance is noted for eight datasets when the case-level information is considered against MTM-CW. Comparing the results of MTM-CW^{ca} with the MMT-CW$^{w/o}$, a performance gain is noted for eleven datasets. The Softmax-based model (MTM-CW^{soft}), again yields the low F1-score against the CRF-based model (MTM-CW). This is also observed in section 3.3.2. This concludes that CRF is more suitable for the sequence labelling problem compared to Softmax.

Datasets	MTM-CW	MTM-CW$^{w/o}$	MTM-CW^{ca}	MTM-CW^{soft}
AnatEM	**87.50**	86.94	87.37	86.36
BC2GM	81.57	81.29	**81.66**	80.04
BC4CHEMD	**89.24**	87.44	89.13	86.88
BC5CDR	88.54	88.11	**88.64**	87.39
BioNLP09	88.52	**89.31**	88.61	88.18
BioNLP11EPI	**85.36**	85.01	85.04	84.16
BioNLP11ID	87.19	**88.16**	87.76	87.28
BioNLP13CG	**84.94**	84.61	84.86	84.00
BioNLP13GE	80.91	**82.28**	80.16	80.49
BioNLP13PC	89.16	89.04	**89.26**	88.37
CRAFT	**85.23**	83.44	85.04	82.86
Ex-PTM	81.72	**82.40**	81.50	80.64
JNLPBA	72.10	72.02	**72.21**	70.31
linnaeus	88.12	88.69	**88.74**	88.33
NCBI-disease	85.07	85.12	**85.56**	84.36
Average	85.01	84.92	**85.04**	83.98

Tab. 3.8: Comparison between the results of different variants of the proposed model.

3.5.2 Statistical Analysis of MTM-CW

To statistically evaluate the results obtained by the proposed MTM-CW [97] models, the Friedman test is performed [85]. Figure 3.5 shows the post-hoc Conover Friedman test where it can be seen that the difference between results produced by all the models is statistically significant with confidence level of 99.9% ($p < 0.001$). The proposed model (MTM-CW) is only statistically significant with the MTM-CW^{soft} (the proposed MTM-CW with Softmax function). The results of different variants of MTM-CW are statistically not significant with each other.

The statistical analysis is also extended with the pairwise comparison of different models to see which model is statistically better than the others. The graphical representation of the pairwise comparison is shown in Figure

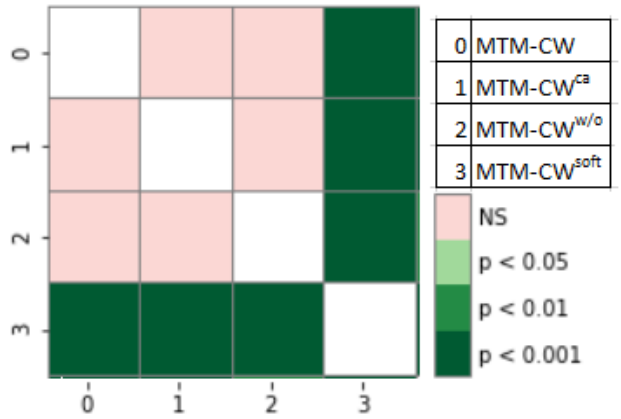


Fig. 3.5: post hoc pairwise analysis with Nemenyi of Friedman test for MTM-CW

Fig. 3.6: Graphical representation of the Friedman test for MTM-CW. The arrows show models that are statistically significant with each other. Models are shown according to their ranks starting best model from left to right.

3.6. It can be seen that MTM-CW^{soft} is statistically worse compared to the other models. The MTM-CW^{ca} is found statistically better than the rest of the approaches on its right side.

The statistical analysis of MTM-CNN vs MTM-CW is summed up in Figure 3.7. It can be noticed that MTM-CW^{ca} is statistically significant with MTM-CNN and its variants. The MTM-CW is also statistically significant against MTM-CNN and its variants except for MTM-CNN$^{\bigstar in}$.

The comparison of MTM-CNN and MTM-CW based on the Friedman test ranks is shown in Figure 3.8. The arrow lines show that the models are statistically significant with each other or group of models. It can be noticed that MTM-CW^{ca} and MTM-CW are statistically better than the rest of the

approaches. More specifically, MTM-CW^{ca} is statistically superior to the rest of the models.

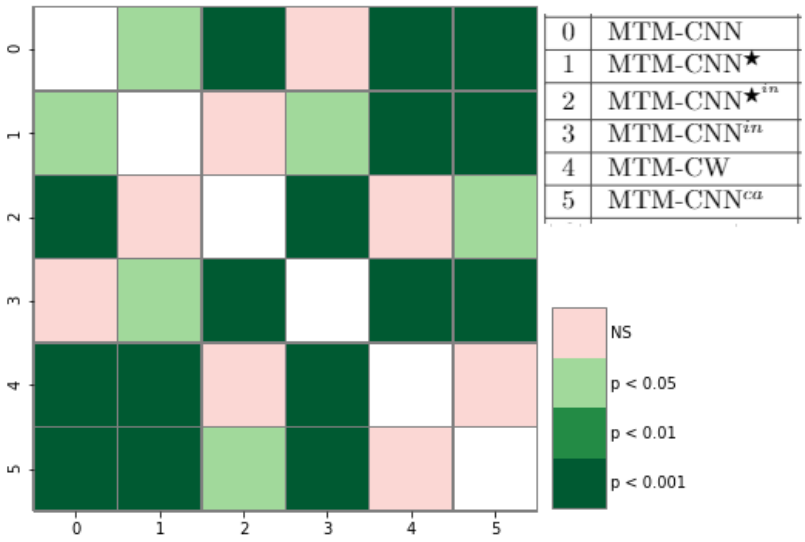


Fig. 3.7: post hoc pairwise analysis with Nemenyi of Friedman test for MTM-CNN vs MTM-CW

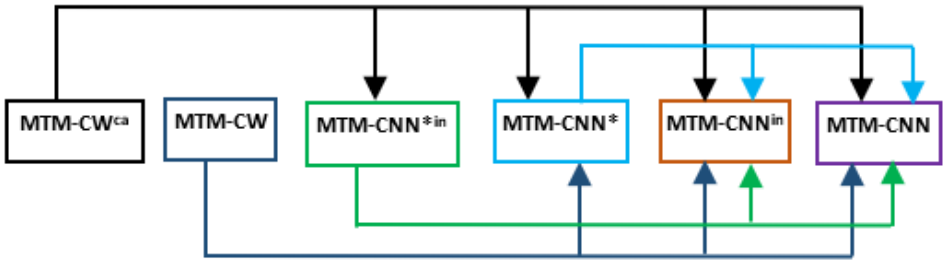


Fig. 3.8: Graphical representation of the Friedman test for MTM-CW vs MTM-CNN. The arrows show models that are statistically significant with each other. Models are shown according to their ranks starting best model from left to right.

4. BIONER USING TRANSFER LEARNING

The results of chapter 3 have demonstrated that MTM can achieve significant improvements over STM in BioNER. However, for some datasets the performance has dropped, when trained with MTM. This indicates that some data may confine the learning process of the MTM. Additionally, the results of MTM-CNN$^{\bigstar^{in}}$ and MTM-CNN^{in} (section 3.3.3) advise that there is a hierarchical learning from standard representations of various tasks in MTM. It is not easy to find the underlying hierarchical relationship between the auxiliary task and the main task.

This chapter seeks to overcome the MTM limitations using transfer learning. In transfer learning, a model is trained (usually partially) on an auxiliary task which is then re-used on the main task. Similarly, the model can be trained on a source domain which can then be re-used on the target domain. The deep learning models are generally initialized with random weights before training. However, for many tasks, these models fail to produce the desired results due to the absence of prior knowledge for that task. In other words, it is better to guide the new model by initializing it with pre-trained weights. This supervised signal also shortens the training time of the learning algorithm. Transfer learning, however, can produce poor results when negative knowledge is transferred [98]. The transfer learning approach has been used in many deep learning tasks, e.g., many natural language processing tasks have benefited from the BERT (Bidirectional Encoder Representations from Transformers) model using the mentioned approach [99, 100]. Howard and Ruder [101] fine-tuned a pre-trained long short-term memory (LSTM) language model for new tasks in different ways. This included unfreezing

various layers step-by-step and also introducing distinct learning rates for different layers. Radford et al. [102] fine-tuned a pre-trained transformer-based language model for task-specific input transformations during an MTL approach. Oquab et al. [103] trained a model on a huge dataset to extract the features for the dataset with few training instances. Al-Stouhi and Reddy [104] empirically showed that the performance of the model can be leveraged using transfer learning for imbalanced labels' dataset. Yang et al. [105] used a pre-trained POS tagging model for word segmentation. Zoph et al. [106] used high-resource language pair to pre-train a machine translation model, which was then applied to a low-resource language pair.

In this chapter, we fine-tune the pre-trained MTM for a specific target dataset. The MTM is trained on various datasets to learn common features which is the starting point for a STM ($MTM \rightarrow STM$). This allows a model to learn in a guided way and at the same time learn the features specific to the target dataset.

4.1 Multi-task Model with Transfer Learning

The proposed approach uses multi-task learning with transfer learning. The MTM model is similar to the one proposed in [53, 107] but is extended with task-specific BiLSTM as shown in Figure 4.1. The MTM uses word and character representations of the sentence and is trained on all 15 datasets that are mentioned in Section 2.2. It is used as a base model and is reused as the starting point of an STM ($MTM \rightarrow STM$). The auxiliary task involves the training MTM on various datasets whereas the main task is the training of a STM for each specific dataset that is initialized by the pre-trained MTM. For transfer learning, the auxiliary task and the main task model remain the same. More specifically, the base MTM is fine-tuned for a specific dataset ($MTM \rightarrow STM$). In particular, neither new layer(s) is introduced or cut-off during fine-tuning of the base MTM. In fact, adding new task-specific layer(s)

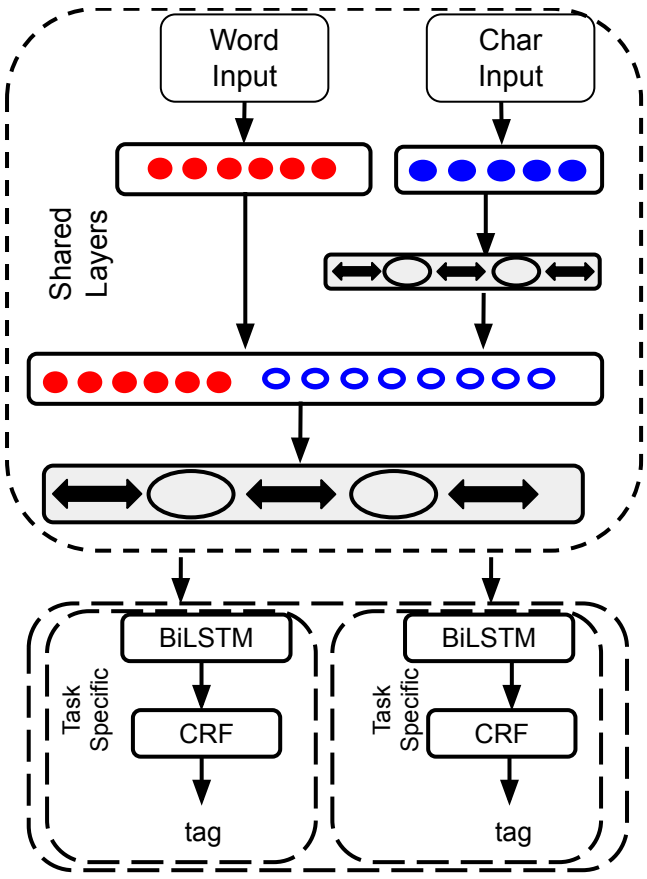


Fig. 4.1: Our proposed model used for fine-tuning ($MTM \rightarrow STM$).

and initializing them with random weights could degrade the performance of the model during fine-tuning as the model can face a lack of guidance for the new task [108].

Indeed, our purpose is to keep the general representation of the features that are learned from different tasks in the training phase of MTM [109]. The purpose is to learn the common features among all the datasets and so generalize the parameters of the model, which then will be further fine-tuned for the specific target dataset; the idea is to move from a generalized model (MTM) to a specialized model (STM). Yosinski et al. [11] performed

experiments to compare the feature extractor and the fine-tuning techniques. They found that, for the feature extractor, the performance of the main task model depends on where the layers are cut. While investigating the fine-tuning method, they found it useful without any strict constraint of layers cut at any level of the base model.

4.1.1 Experimental Settings

Most of the experimental setups is the same as indicated in subsection 3.2. The character embedding is initialized randomly whereas for word embedding Wiki-PubMed-PMC [55] is used. The experiments are conducted for three variants of the proposed method where MTM is trained for a specific number of epochs and then transfer learning is applied, where the MTM is fine-tuned for a specific dataset. In the first experiment ($MTM^{10}\rightarrow STM$), the MTM^{10} is trained for ten epochs to learn the standard features representation of various tasks and then fine-tuned on a specific dataset. In the second experiment ($MTM^{20}\rightarrow STM$), the MTM^{20} is trained for twenty epochs followed by fune-tuning it for a specific dataset. In the last experiment, the MTM^{cmp} is trained until the early stop occurs or the total number of epochs is exhausted (the early stop is used for earlier experiments as well) and then fine-tuned for a specific dataset ($MTM^{cmp}\rightarrow STM$). The reported F1-scores represents average score of 10 runs.

4.2 Results and Discussions for Fine-tuned MTM ($MTM\rightarrow STM$)

Table 4.1 summarizes the results of the experiments. The best F1-score is shown with **boldface** text while *Italics* font style shows the second best F1-score. In the table, a performance increment can be observed for MTM in most of the datasets compared to STM; whereas, a decrease of F1-score for BC2GM, BC4CHEMD, BC5CDR, and CRAFT is also noticed. This

Datasets	STM	MTM	$MTM^{10}\rightarrow STM$	$MTM^{20}\rightarrow STM$	$MTM^{cmp}\rightarrow STM$
AnatEM	86.74	87.59	*87.91*	**88.01**	**88.01**
BC2GM	81.77	81.69	*82.15*	**82.21**	82.06
BC4CHEMD	90.40	89.01	89.99	**90.47**	*90.46*
BC5CDR	88.54	88.40	88.88	*89.00*	**89.13**
BioNLP09	87.89	**89.03**	88.54	*88.71*	88.56
BioNLP11EPI	83.16	85.28	85.34	**85.57**	*85.44*
BioNLP11ID	86.31	87.53	87.63	*87.82*	**87.94**
BioNLP13CG	83.13	84.93	84.95	**85.23**	*85.19*
BioNLP13GE	76.47	**80.39**	80.11	80.10	*80.25*
BioNLP13PC	87.76	89.23	*89.34*	89.27	**89.36**
CRAFT	84.77	84.26	84.92	**85.36**	*85.09*
Ex-PTM	74.04	**82.14**	81.76	*82.09*	81.82
JNLPBA	72.25	*72.80*	**73.00**	72.13	71.94
LINNAEUS	87.62	88.47	*88.82*	88.27	**88.84**
NCBI-disease	84.91	*86.24*	*86.24*	85.94	**86.29**
Average	83.72	85.13	85.31	*85.35*	**85.36**

Tab. 4.1: Results comparison of our different fine-tune approaches of MTM ($MTM{\rightarrow}STM$).

indicates that MTM does not always manage to enhance the performance of the STM.

From Table 4.1, it can be seen that fine-tuning, ($MTM^{10}\rightarrow STM$), produces an increase in performance for all datasets compared to STM, except for BC4CHEMD. BC4CHEMD is a single entity dataset that contains a large number of chemical entities reducing the sparsity presented in the data. Therefore, ($MTM^{10}\rightarrow STM$) learns features more distinguishable compared to the MTL approach; furthermore, comparing it with the MTM, a performance increase is observed for all datasets except for BioNLP09, BioNLP13GE, and Ex-PTM. Observing the results of STM and MTM for these three datasets

shows a substantial performance gain with MTM approach. This indicates that it is difficult to learn complex feature from these datasets independently in a STM and can be learn in a MTM. The training of MTM^{10} extracted useful features for these datasets but might start to forget such features, or the noise presented in these datasets might have caused a performance drop.

Continuing the experiments, the MTM model trained for 20 epochs, (MTM^{20}), is fine-tuned ($MTM \overset{20}{\rightarrow} STM$) for each specific dataset. Comprehensively, it can be noticed that JNLPBA is the only dataset, out of 15 datasets, where ($MTM \overset{20}{\rightarrow} STM$) fails to show an increase in performance compared to the STM, and in fact, a small decrease in performance is noticed. An overall performance increment can be observed for nine datasets compared with MTM, while for six datasets a decrease in performance is noticed. Furthermore, comparing with ($MTM \overset{10}{\rightarrow} STM$) approach, a performance improvement can be seen for ten datasets, while performance decrease is noticed for five datasets. Comparing F1-score with MTM, the ($MTM \overset{20}{\rightarrow} STM$) fails to achieve better results for some protein datasets (BioNLP09, BioNLP13GE, Ex-PTM) while for BioNLP11PEI a performance gain is noticed. In the JNLPBA dataset, which contains a large number of examples for the protein class, also shows a decrease in performance. The decrease in performance for the LINNAEUS dataset might be due to the absence of a sufficient number of examples for the entity class. This suggests that these datasets are more feasible with the MTL approach. It can also be seen that ($MTM \overset{20}{\rightarrow} STM$) has achieved better results compared to the STM and MTM on BC4CHEMD. It is also worth noting that the F1-score of MTM on BC4CHEMD is worse than the STM one.

In a third experiment, a fully trained MTM (MTM^{cmp}) is fine-tuned for each dataset. It is observed that ($MTM \overset{cmp}{\rightarrow} STM$) improved the results for 11 datasets compared to the MTM whereas comparing it with the STM, the JNLPBA is the only dataset (out of 15 datasets) where a performance decrease is noticed. Furthermore, comparing it with ($MTM \overset{10}{\rightarrow} STM$), the

method yields a performance increment for thirteen datasets whereas comparing with ($MTM \xrightarrow{20} STM$), a performance increment is noticed for seven datasets while for AnatEM the performance does not change. Furthermore, likewise in ($MTM \xrightarrow{20} STM$) and ($MTM \xrightarrow{10} STM$), the ($MTM \xrightarrow{cmp} STM$) performs worse for Protein datasets (BioNLP09, BioNLP13GE, Ex-PTM, and JNLPBA) compared to the MTM. However, unlike in ($MTM \xrightarrow{20} STM$), a performance improvement can be seen for LINNAEUS and NCBI datasets compared to MTM. The ($MTM \xrightarrow{cmp} STM$) achieves the best F1-score for BC4CHEMD where MTM performs worse with respect to STM. Comparing it with the ($MTM \xrightarrow{10} STM$) method, the JNLPBA is the only dataset where ($MTM \xrightarrow{cmp} STM$) fails to achieve a performance gain. Moreover, comparing with the ($MTM \xrightarrow{20} STM$), a performance decrease is noticed for BC2GM, BioNLP09, BioNLP11EPI, BioNLP13CG, CRAFT, Ex-PTM, and JNLPBA whereas for AnatEM and BC4CHEMD the difference is negligible. It is speculating that the reason for the drop in F1-score is related to the shared layer in ($MTM \xrightarrow{cmp} STM$) which has learned features that became more task specific and therefore favoring only specific datasets.

The proposed method, ($MTM \rightarrow STM$), is compared with the STMs presented in [53] and in section 3.1 and is shown in Table 4.2. The table illustrates that the proposed method outperforms the previous approaches [53, 95], explaining their proposed STMs may have lacking the distinct features. In the case of fine-tuned models ($MTM \rightarrow STM$), the common features are transferred from pre-trained MTM.

The results of the proposed method are compared with the state-of-the-art MTMs as well and are represented in Table 4.3, where it can be observed that our proposed fine-tuned models ($MTM \rightarrow STM$ [110]) have shown an increase in performance compared to other methods.

The proposed MTM-CW (section 3.4) has increased in performance for a single dataset while proposed MTM-CNN 3.1 has shown a performance gain for only two datasets compared to all fine-tuned models.

Datasets	STM [53]	STM [95]	$MTM^{10}{\rightarrow}STM$	$MTM^{20}{\rightarrow}STM$	$MTM^{cmp}{\rightarrow}STM$
AnatEM	85.00	85.89	*87.91*	**88.01**	**88.01**
BC2GM	80.00	80.9	*82.15*	**82.21**	82.06
BC4CHEMD	88.75	88.6	89.99	**90.47**	*90.46*
BC5CDR	86.96	85.66	88.88	*89.00*	**89.13**
BioNLP09	84.22	87.03	88.54	**88.71**	*88.56*
BioNLP11EPI	77.67	81.48	85.34	**85.57**	*85.44*
BioNLP11ID	74.60	83.21	87.63	*87.82*	**87.94**
BioNLP13CG	81.84	81.27	84.95	**85.23**	*85.19*
BioNLP13GE	69.30	73.36	*80.11*	80.10	**80.25**
BioNLP13PC	85.46	86.33	*89.34*	89.27	**89.36**
CRAFT	81.20	83.84	84.92	**85.36**	*85.09*
ExPTM	67.66	72.70	81.76	**82.09**	*81.82*
JNLPBA	72.17	**74.48**	*73.00*	72.13	71.94
linnaeus	86.94	87.38	*88.82*	88.27	**88.84**
NCBI	83.92	84.11	*86.24*	85.94	**86.29**
Average	80.38	82.42	85.31	*85.35*	**85.36**

Tab. 4.2: Results comparison of proposed $MTM{\rightarrow}STM$ method with state-of-the-art STMs

4.2.1 Statistical Analysis of $MTM{\rightarrow}STM$

The results are evaluated using Friedman's statistical test as presented in Figure 4.2. The figure shows that all the variants of $MTM{\rightarrow}STM$ have produced statistically significant results against the counterpart STM and MTM. The results are statistically significant with previously mentioned approaches, MTM-CNN [95] and MTM-CW [97]. Nevertheless, the $MTM{\rightarrow}STM$ have not generated significant results with each other.

The models are also shown according to their Friedman statistical ranks and are given in Figure 4.3. For the readability purpose, large blue rectangle is used to group all the models that are found statistically significant based on

Datasets	MTM-CNN [95]	MTM-CW [97]	$MTM^{10}{\rightarrow}STM$	$MTM^{20}{\rightarrow}STM$	$MTM^{cmp}{\rightarrow}STM$
AnatEM	86.99	87.50	*87.91*	**88.01**	**88.01**
BC2GM	80.82	81.57	*82.15*	**82.21**	82.06
BC4CHEMD	87.39	89.24	89.99	**90.47**	*90.46*
BC5CDR	87.85	88.54	88.88	*89.00*	**89.13**
BioNLP09	**88.74**	88.52	88.54	*88.71*	88.56
BioNLP11EPI	84.75	85.36	85.34	**85.57**	*85.44*
BioNLP11ID	87.65	87.19	87.63	*87.82*	**87.94**
BioNLP13CG	84.25	84.94	84.95	**85.23**	*85.19*
BioNLP13GE	79.82	**80.91**	80.11	80.10	*80.25*
BioNLP13PC	88.84	89.16	*89.34*	89.27	**89.36**
CRAFT	83.15	*85.23*	84.92	**85.36**	85.09
ExPTM	80.95	81.72	81.76	**82.09**	*81.82*
JNLPBA	**74.05**	72.10	*73.00*	72.13	71.94
linnaeus	87.79	88.12	*88.82*	88.27	**88.84**
NCBI	85.66	85.07	*86.24*	85.94	**86.29**
Average	84.58	85.01	85.31	85.35	**85.36**

Tab. 4.3: Results comparison of $MTM{\rightarrow}STM$ with state-of-the-art MTMs

their results to $MTM^{cmp}{\rightarrow}STM$, $MTM^{20}{\rightarrow}STM$, and $MTM^{10}{\rightarrow}STM$. The figure presents that all the fine-tuned models have produced higher Friedman statistical scores. The $MTM^{cmp}{\rightarrow}STM$ has generated the highest statistical rank indicating that this model has covered a wider range of features, benefiting datasets suitable for both the MTL and STL approaches. The $MTM^{10}{\rightarrow}STM$ has produced the lowest ranks among the fine-tuned models, which shows that the base model (MTM^{10}) for that experiment has not learned distinct features, and therefore, using that model as a starting point for STM has not benefited the model.

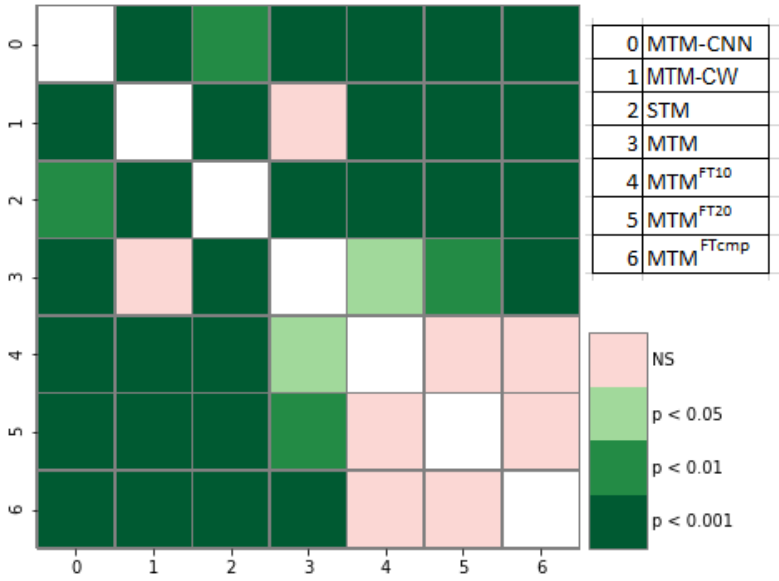


Fig. 4.2: post hoc Conover Friedman test output for $MTM{\rightarrow}STM$.

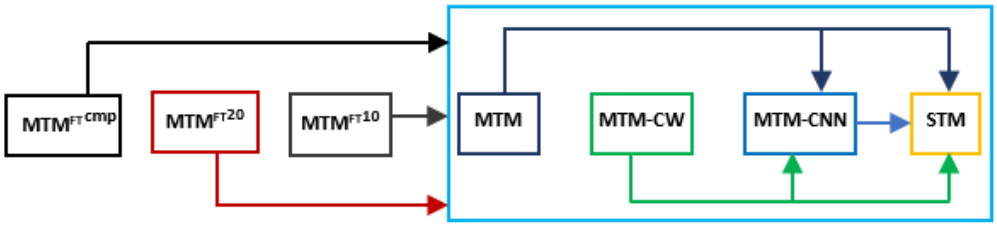


Fig. 4.3: Graphical representation Friedman test ranks produced by $MTM{\rightarrow}STM$. The arrows show models are statistically significant with each other. Models are shown according to their ranks starting best model from left to right.

5. BIONER USING KNOWLEDGE DISTILLATION

The transfer learning technique has shown a performance improvement for those datasets that do not have enough training examples. However, transfer learning also faces limitations, for instance, catastrophic forgetting or catastrophic interference problem [84]. In catastrophic forgetting, the deep learning model starts to forget what it has learned from the previous domain. Failing to remember previously learned source information happens even if both source and target domains are heterogeneous [111]. It is also an empirical dilemma to choose the number of new layers in the base or target model that is to be used on the target dataset. Similarly, the layers that need to be frozen in the base or target model as it is applied on the target dataset. As a result, the transfer learning approach is not always a feasible solution for transferring the previous knowledge to a new task. Another issue related to the deep learning models is their complex structure. In many fields, deep learning models have produced state-of-the-art results, but the structure of these models is very complex and requires extensive computational power for training. Sutskever et al. [112] proposed a model that comprised of 4-layers of LSTMs and each layer had 1000 hidden units. Similarly, the model presented by Zhou et al. [113] had multi-level LSTMs and each layer comprised of 512 hidden units. These deep learning models have millions of parameters and training such models is challenging. These cumbersome models also require more storage space, making them unsuitable to deploy with real-time data. One example is to use them on a cell phone where limited storage and computational power is available. As a result, it is necessary to compress these complex models while preserving the generalization they have learned.

In other words, without jeopardising the performance of these deep learning models. In this case, the knowledge distillation approach is utilized, in which the cumbersome model is compressed into the simple model which makes it feasible to set up in the end devices [114]. This chapter proposes the distillation knowledge approach to leverage the performance of the deep learning models. Instead of compressing the model, this research aims to maximize the efficiency and performance of models. For this reason, this chapter presents different approaches to boost the performance of the deep neural network models.

5.1 Proposed Knowledge Distillation Approach

The proposed knowledge distillation approach [115] is shown in Figure 5.1. The teacher model is an MTM that uses the word and character input representations of the sentences. The upper layers, shown in the black round rectangle, of the MTM are shared among all datasets. The bottom layers, represented by the red round rectangle, are dataset-specific, whereas the Softmax function is used for output labelling. Training jointly on the related tasks allows MTM to learn common features among different tasks by using shared layers [60]. Training related tasks together helps the model to optimize its parameters for different tasks. This lowers the chances of overfitting for any specific task. The task-specific layers learn features that are more related to the current task.

Such behaviour of MTM can also be transferred to the student model with the help of knowledge distillation approach. Thus, MTM (right side of Figure 5.1) is a teacher model used in the later experiments presented in this chapter (section 5.2). Furthermore, the aim of this research is to perform knowledge distillation at the token level. For this reason, the proposed approach employs Softmax function which produces probability distribution at the token level. The token level knowledge distillation is not possible with conditional

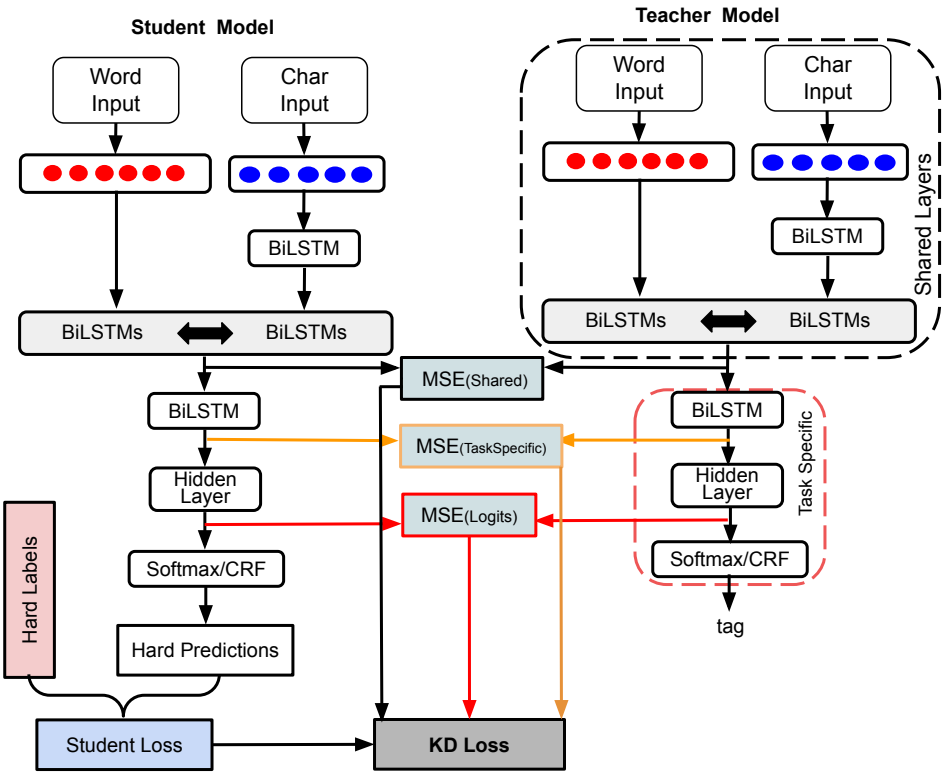


Fig. 5.1: Proposed Knowledge Distillation Approach (colored circles show embedding)

random fields (CRFs) as they predict the labels for the whole sequence. The CRF-based model labels the sequence by considering the association between neighboring labels. This confines the knowledge distillation from the teacher models [116]. To verify this hypothesis, two different teacher MTMs are utilized, one using CRF at the output layer and the other using the Softmax function.

The student model is in fact a counterpart STM of the MTM. As such, STM is a student model but trained without knowledge distillation approach. Therefore, the structure of both models is same. The proposed approach utilizes different layers of MTM for knowledge distillation. Deep learning model

comprehends complex features from generic level to abstract level, layers by layers. Inferring that, different intermediate layers of the teacher model are used for knowledge distillation. This includes a shared BiLSTM layer, a task specific BiLSTM layer, and a hidden layer logits (refer Figure 5.1). As such, the student model is trained in a guided way at different levels, speculating this will increase the performance of the student model. The layers' knowledge is integrated from logits's layer to different layers' step by step in different experiments to observe the effects of each integration. The logits (input to the Softmax layer [117]) carry values that can range in $[-\infty, +\infty]$, are unmasked information, and therefore the features representation at this level is more beneficial and refined. The Softmax function uses logits to produce a probability distribution of class labels, which causes to loss of hidden knowledge present in the logits' layer. Considering that, this chapter also contains the experiments, where a student model is trained on the logits of the MTM (section 5.2.1). However, to validate this hypothesis, the knowledge distillation is also performed at the Softmax layer using soft labels where the output probability distribution of the teacher model is softened by a tunable parameter τ.

This chapter also explores the advantages of ensemble methods in two different ways. In the first way, logits (input to the Softmax layer [117]) of different MTMs are combined to train the student model. The MTMs used in the ensemble approach have the same architecture, but initialized with different seed values, which causes MTMs to have different trained weights. The combined logits are averaged which are then used to distill knowledge to the student model. In the second method of the ensemble approach, logits of the CRF-based and Softmax-based MTMs are averaged to train the student model (SM). The rationale behind this approach is that different MTMs learn different features set, and therefore, the SM is trained on a wider range of the features.

During the training, the SM considers the true labels as well as the

distillation loss, which depends by matching the outputs of the teacher model (MTM). For the intermediate layers' knowledge distillation loss, mean-squared-error (MSE) is calculated which aims to minimize the loss between the student predictions and the teacher predictions at different layers. On the contrary, when soft labels are considered then cross entropy loss is used.

5.2 Experimental Settings

As a first approach, the MTM, shown in the right side of Figure 5.1, is trained separately. This MTM is then used to distill the knowledge to the student model (SM). When the knowledge distillation is performed using the logits of the hidden layer, Equation 5.1 is used to compute the loss of the SM. The knowledge distillation loss is the MSE of the teacher and the student logits. Here, x represents the input, W represents SM's parameters, $\mathcal{H}$ is the cross-entropy loss whereas y corresponds to the true labels. The σ is the Softmax function applied on the teacher logits, z_s, and student logits, z_t. The α and β are hyperparameters to quantify each loss.

The experiments are conducted for different values of α i.e.,$[0, 0.5, 1]$ whereas $\beta = 1 - \alpha$. The hyperparameter tuning for α and β is not done, instead the values are selected in a simple straightforward way. If $\alpha = 0$, then the SM learns with only distillation loss, while keeping $\alpha = 0.5$ makes equal use of both student loss and distillation loss. Finally, $\alpha = 1$ only allows the SM to consider the student loss. It should be noted that for $\alpha = 1$, the SM becomes a single task model (STM). The SM, however, still considers the logits of the teacher model during its training phase. This helps the SM to learn and modify the weights of layers during the back propagation phase.

When the knowledge distillation is carried out at the task specific BiLSTM layer along with the logits of the hidden layer, the loss is computed using Equation 5.2. The new parameter, $\gamma = [0, 0.5, 1]$, weighs the matching loss at the task specific BiLSTM layer. The hyperparameters tuning is per-

formed to select best value for α, β and γ. When knowledge of the shared BiLSTM is incorporated along with the above layer's knowledge, then loss is calculated using Equation 5.3. The MSE error of the task specific BiLSTM is controlled using the new parameter, $\kappa = [0, 0.5, 1]$. For Equation 5.3, hyperparameter tuning is performed for $\beta, \gamma,$ and κ to select the best value from $[0, 0.5, 1]$ for each parameter. While α is kept constant to 1. When the knowledge distillation is carried out using soft labels, the Equation 5.4 is used to calculate the loss. The α and β parameters retained to 0.5 while τ and is fine-tuned for each dataset. The F1-scores reported are based on the 10 runs.

$$\mathcal{L}(x; W) = \alpha \cdot \mathcal{H}(y, \sigma(z_s, z_t)) + \beta \cdot MSE_{logits} \tag{5.1}$$

$$\mathcal{L}(x; W) = \alpha \cdot \mathcal{H}(y, \sigma(z_s, z_t)) + \beta \cdot MSE_{logits} + \gamma \cdot MSE_{TaskSpecific} \tag{5.2}$$

$$\mathcal{L}(x; W) = \alpha \cdot \mathcal{H}(y, \sigma(z_s, z_t)) + \beta \cdot MSE_{logits} + \gamma \cdot MSE_{TaskSpecific} + \kappa \cdot MSE_{Shared} \tag{5.3}$$

$$\mathcal{L}(x; W) = \alpha \cdot \mathcal{H}(y, \sigma(z_s, z_t)) + \beta \cdot \mathcal{H}(y, \sigma(z_s/\tau, z_t/\tau)) \tag{5.4}$$

5.2.1 Knowledge Distillation Using Logits of the Teacher Model

Table 5.1 presents the results comparison of the SMs^{ψ} with STM and MTM (a teacher model). The SMs^{ψ} have outperformed for most of the datasets compared with the STM except for BC4CHEMD and CRAFT. Observing the results of MTM for BC4CHEMD and CRAFT, it is noticed that MTM is not able to improve the results for these datasets. This might be the reason that the SMs^{ψ} are not able to learn sufficient knowledge from the MTM. It

Datasets	STM	MTM	SMs^{Ψ}		
			$\alpha = 0$	$\alpha = 0.5$	$\alpha = 1$
AnatEM	86.63	86.75	87.32	*87.35*	**87.46**
BC2GM	*81.12*	79.82	80.98	81.09	**81.35**
BC4CHEMD	**90.19**	86.71	89.35	89.44	*89.48*
BC5CDR	88.06	87.38	87.84	*88.12*	**88.19**
BioNLP09	87.49	88.35	**88.77**	88.59	*88.72*
BioNLP11EPI	82.66	*84.49*	84.33	**84.57**	83.75
BioNLP11ID	85.65	**87.25**	*86.99*	86.95	86.41
BioNLP13CG	82.11	**83.91**	83.07	*83.49*	82.91
BioNLP13GE	75.51	**79.67**	77.60	*77.92*	76.60
BioNLP13PC	87.03	**88.37**	88.05	*88.17*	87.72
CRAFT	**84.32**	82.14	83.29	84.00	*84.16*
ExPTM	73.28	**80.80**	76.09	*76.43*	75.16
JNLPBA	70.82	70.37	71.35	**72.14**	*71.58*
linnaeus	87.66	88.34	**88.51**	*88.50*	88.23
NCBI	84.15	84.74	*84.91*	**85.03**	84.44
Average	83.11	*83.94*	83.90	**84.12**	83.74

Tab. 5.1: Results comparison of the SM^{ψ} trained with logits of the Softmax-based teacher MTM.

is still worth noting that SMs^{ψ} have improved the results by 2.5% for these two datasets compared with the MTM, showing the benefits of the knowledge distillation approach. The $SM^{\psi}(\alpha = 0.5)$ trained with both, the student loss and the distillation loss, has been found more effective, yielding an average F1-score of 84.12 followed by the $SM^{\psi}(\alpha = 0)$ trained with only knowledge distillation loss resulting an average F1-score of 83.90.

The statistical analysis is also performed on the results of Table 5.1 using Friedman test, where the results are found statistically significant with $p < 0.05$. Figure 5.2 depicts the graphical representation of the model based

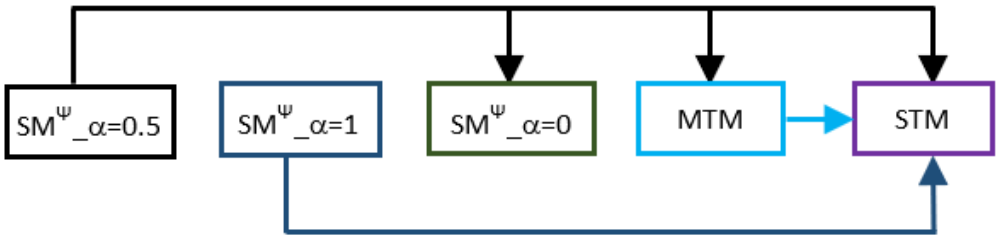


Fig. 5.2: Graphical representation of the Friedman test for Table 5.1. The arrows show models that are statistically significant with each other. Models are shown according to their ranks starting from best model from left to right.

on the ranks generated by the Friedman test. For better understanding, different colors are used for the rectangles and arrows, where the arrow has the same color as of the rectangle to which it belongs to. The $SM^{\psi}(\alpha = 0.5)$ produces the best results and is statistically significant w.r.t to all other models except for the SM^{ψ} with $(\alpha = 1)$. This is the only SM^{ψ} that generates statistically significant results w.r.t to its teacher model (MTM). The results of SM^{ψ} with $(\alpha = 0)$ are found worse among the SMs^{ψ} that is unable to produce statistically significant results against STM.

5.2.2 Knowledge Distillation Using an Ensemble Approach

In machine learning, ensemble approaches have been found more effective than the single machine learning model approach. Deep learning models are nonlinear, and therefore produce a different set of weights each time a single model is trained. The initial weights can also cause different predictions, resulting in high variance in the predictions. In order to reduce such variance, various neural network models can be trained and combined to generate a single prediction. Furthermore, the single model might not necessarily learn distinctive features present in the data. This limitation can be tackled in the ensemble approach where feature representations from different models are combined in a single ensemble model. The predictions from different models

can be combined using different methods including voting, average, weighted voting/average schemes.

This section introduces two types of ensemble methods where different predictions are combined using a weighted average scheme. In this scheme, predictions from different models are averaged and combined according to their estimated performance. In this section feature representations at the logits layer are combined from various models, which enhances the learning ability of the SM to learn from a wide range of feature representations. The MTMs used in the ensemble approach have the same architecture, but they are initialized with different seed values, resulting in different predictions. In the first proposed approach, the SMs^{ϕ} use the weighted averaged logits from different MTMs to train the SM^{ϕ}. The averaged logits of these MTMs are used to train the SM^{ϕ}. In the second ensemble approach, the weighted averaged logits of Softmax-based MTMs and CRF-based MTMs are used to train the $SM^{\S}$.

Table 5.2 represents the first ensemble approach where the results are improved over the previous single teacher distillation approach. The performance improvement is also noted for the BC4CHEMD and CRAFT datasets, as these datasets have shown a performance drop, when trained with the logits of a single MTM (section 5.2.1). The increase in performance is also noted for BioNLP11ID, BioNLP13CG, and BioNLP13PC against the MTM.

The results of the second ensemble approach are presented in Table 5.3. The logits of CRF-based MTM (MTM^{CRF}) are used in conjunction with the logits of Softmax-based MTM, where the average sum of both logits are used to train the student model. This makes both models a teacher model for the student model, and therefore, the given table also contains the results of CRF-based MTM (MTM^{CRF}) for comparison. The CRF-based MTM (MTM^{CRF}) has shown the best F1-score for most of the datasets even compared with the Softmax-based MTM. However, when knowledge distillation is performed using the same MTM^{CRF} and MTM, the performance of the

Datasets	STM	MTM	SMs^{ϕ}		
			$\alpha = 0$	$\alpha = 0.5$	$\alpha = 1$
AnatEM	86.63	86.75	*87.94*	**88.04**	86.99
BC2GM	81.12	79.82	*81.93*	**82.19**	81.02
BC4CHEMD	*90.19*	86.71	88.87	**90.58**	*90.18*
BC5CDR	88.06	87.38	*88.54*	**88.80**	88.35
BioNLP09	87.49	88.35	**89.44**	*89.08*	*87.28*
BioNLP11EPI	82.66	84.49	*84.95*	**85.00**	82.58
BioNLP11ID	85.65	87.25	**87.71**	*87.63*	85.37
BioNLP13CG	82.11	*83.91*	83.78	**84.06**	82.40
BioNLP13GE	75.51	**79.67**	77.93	*78.03*	75.80
BioNLP13PC	87.03	88.37	**88.80**	*88.76*	87.22
CRAFT	*84.32*	82.14	83.80	**84.96**	84.10
ExPTM	73.28	**80.80**	*77.23*	76.78	73.27
JNLPBA	70.82	70.37	*71.96*	**72.74**	71.18
linnaeus	87.66	88.34	**89.66**	*89.57*	87.96
NCBI	84.15	84.74	**86.20**	*86.08*	84.25
Average	83.11	83.94	*84.58*	**84.82**	83.20

Tab. 5.2: Results comparison of the SM^{ϕ} trained with logits of the ensemble teacher Softmax-based MTMs.

$SMs^{\S}$ does not improve for most of the datasets. The reason could be that the CRF based model labels the sequence globally considering the association between neighboring labels. Therefore, this limits the distilling knowledge from the teacher models [116]. This might be the possible reason of the performance degradation of the corresponding SMs. Comparing the $SMs^{\S}$ with MTM, the results are improved for most of the datasets. The $SM^{\S}(\alpha = 0)$ shows a performance gain for 10 datasets compared with the MTM. The $SM^{\S}(\alpha = 0.5)$ yields result improvement for 12 datasets while $SM^{\S}(\alpha = 1)$ gives the worse performance with a performance gain for only 6 datasets.

Datasets	STM	MTM	MTM^{CRF}	$SMs^{\S}$		
				$\alpha = 0$	$\alpha = 0.5$	$\alpha = 1$
AnatEM	86.63	86.75	*87.59*	87.56	**87.72**	87.32
BC2GM	*81.12*	79.82	**81.69**	80.95	*81.55*	81.27
BC4CHEMD	**90.19**	86.71	89.01	89.22	90.14	*90.20*
BC5CDR	88.06	87.38	*88.40*	88.16	**88.46**	88.31
BioNLP09	87.49	88.35	**89.03**	**88.84**	***88.84***	88.19
BioNLP11EPI	82.66	84.49	**85.28**	84.40	*84.58*	83.88
BioNLP11ID	85.65	87.25	**87.53**	*87.48*	87.26	86.52
BioNLP13CG	82.11	*83.91*	**84.93**	83.50	83.67	82.93
BioNLP13GE	75.51	*79.67*	**80.39**	77.98	77.95	76.78
BioNLP13PC	87.03	88.37	**89.23**	88.27	*88.42*	87.78
CRAFT	84.32	82.14	84.26	83.67	**84.63**	*84.51*
ExPTM	73.28	*80.80*	**82.14**	76.52	76.23	74.80
JNLPBA	70.82	70.37	**72.80**	71.03	*72.14*	71.59
linnaeus	87.66	88.34	88.47	**89.16**	*88.97*	88.28
NCBI	84.15	84.74	**86.24**	*85.57*	85.31	84.85
Average	83.11	83.94	**85.13**	84.15	*84.39*	83.81

Tab. 5.3: Results comparison of the $SM^{\S}$ trained with the logits of the Softmax-based MTM and CRF-based MTM (MTM^{CRF}).

The statistical results reported in the Table 5.2 are graphically presented in Figure 5.3. The $SM^{\phi}(\alpha = 0.5)$ set up the best rank and produces statistically significant results as compared to the rest of the models including other variants of $SMs^{\phi}(\alpha = 0$ and $\alpha = 1)$. The $SMs^{\phi}(\alpha = 0.5$ and $\alpha = 0)$ achieved better score against the teacher model (MTM) whereas $SM^{\phi}(\alpha = 1)$ does not produce significant results even compared with the STM.

Figure 5.4 illustrates the Friedman test ranks for the models presented in Table 5.3. It can be seen that none of the $SMs^{\S}$ have produced statistically better results compared with the MTM^{CRF}. However, the $SMs^{\S}(\alpha =$

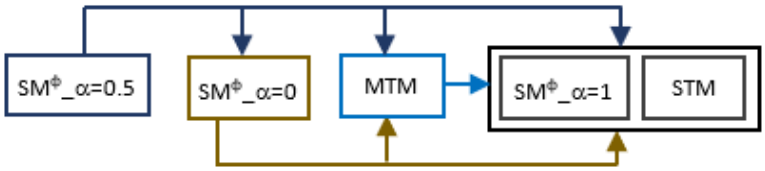


Fig. 5.3: Graphical representation of the Friedman test for Table 5.2. The arrows show models that are statistically significant with each other. Models are shown according to their ranks starting best model from left to right.

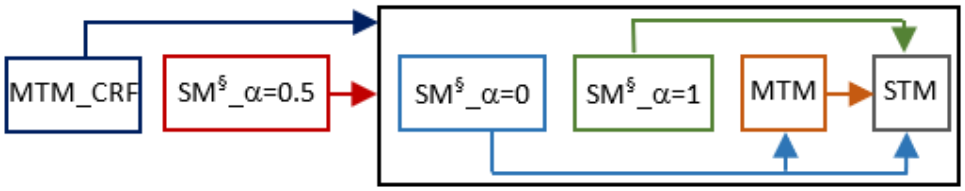


Fig. 5.4: Graphical representation of the Friedman test for Table 5.3. The arrows show models that are statistically significant with each other. Models are shown according to their ranks starting best model from left to right.

0.5 and $\alpha = 0$) have statistically performed well against MTM. Conclusively, all $SMs^{\S}$ are found statistically better than STM.

5.2.3 Knowledge Distillation Using Intermediate Layers of Teacher Model

To extract more knowledge from the teacher model, the knowledge distillation is performed at different levels of the intermediate layers. Different layers learn different feature representations and training SM on the diverse features can be effective. The output of both shared and task specific BiLSTMs (refer to Figure 5.1) is used for knowledge distillation along with the logits of the hidden layer, and the results are depicted in Table 5.4. The MSE is computed for the output of the teacher's BiLSTM and student's BiLSTM. The $SM^{\dagger\dagger}$ with a task specific BiLSTM shows remarkable improvement in all datasets compared with the STM, whereas comparing against MTM, the increase in performance is noted for nine datasets. Likewise, the previously proposed

Datasets	STM	MTM	SM††	SM$^{\dagger\bigstar}$
AnatEM	86.63	86.75	**87.84**	87.83
BC2GM	81.12	79.82	**81.82**	*81.63*
BC4CHEMD	90.19	86.71	*90.52*	**90.57**
BC5CDR	88.06	87.38	**88.89**	*88.83*
BioNLP09	87.49	88.35	**88.39**	*88.37*
BioNLP11EPI	82.66	**84.49**	84.08	*84.39*
BioNLP11ID	85.65	**87.25**	86.64	*86.80*
BioNLP13CG	82.11	**83.91**	*83.69*	83.64
BioNLP13GE	75.51	**79.67**	77.60	*77.99*
BioNLP13PC	87.03	*88.37*	88.29	**88.39**
CRAFT	84.32	82.14	**85.15**	*85.07*
ExPTM	73.28	**80.80**	*75.02*	74.51
JNLPBA	70.82	70.37	**71.77**	*71.69*
linnaeus	87.66	88.34	**89.18**	*88.44*
NCBI	84.15	84.74	*85.51*	**85.58**
Average	83.11	83.94	**84.29**	*84.25*

Tab. 5.4: Results comparison of the proposed Softmax-based SM. $SM^{\dagger\dagger}$ trained with task specific intermediate BiLSTM layer of Softmax-based MTM. $SM^{\dagger\bigstar}$ trained with Shared and task specific intermediate BiLSTM layer of Softmax-based MTM.

SMs, $SM^{\dagger\dagger}$ fails as well to leverage the performance for most of the protein datasets. With the introduction of the shared BiLSTM layer along with the task specific BiLSTM layer ($SM^{\dagger\bigstar}$), the performance of the model increased for some datasets compared to the task specific $SM^{\dagger\dagger}$.

Further analysing the results using Friedman test as shown in Figure 5.5, it is found that results of both SMs ($SM^{\dagger\bigstar}$ and $SM^{\dagger\dagger}$) are not statistically significant with each other. However, they are statistically better than the MTM and STM.

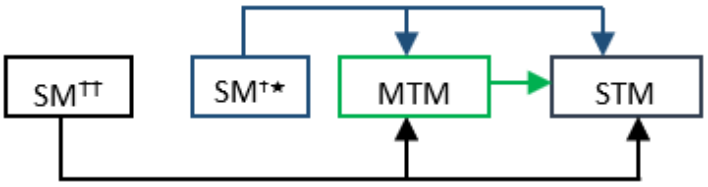


Fig. 5.5: Graphical representation of the Friedman test for Table 5.4. The arrows show models are statistically significant with each other. Models are shown according to their ranks starting best model from left to right.

5.2.4 Knowledge Distillation for CRF-based Student Model

In the above experimentations, the SMs use Softmax at the output layer. This section uses the SM that utilizes CRF at the output layer while the rest of the architecture and approaches remain same. The results used for the CRF-based $SM^{\bigstar}$ are the average F1-score of five runs. For fair comparison, the results of other models (STM and MTM) are also based on five runs. As discussed earlier, the SM is in fact STM but is trained with external knowledge. For this reason, the STM with CRF (STM^{CRF}) at the output layer is selected for the comparison. Table 5.5 depicts the results of the CRF-based $SM^{\bigstar}$ trained on the logits of the CRF-based MTM. The results of the $SMs^{\bigstar}$ drops noticeably against teacher MTM. This demonstrates that the performance of the $SMs^{\bigstar}$ is confined when CRF-based teacher MTM (MTM^{CRF}) is used for knowledge distillation. However, it is quite interesting that the $SMs^{\bigstar}$ still gives high F1-score against the counterpart STM. All the variants of the $SM^{\bigstar}$ demonstrate a performance improvement for ten datasets compared with the STM.

In another experiment, the CRF-based $SM^{\bigstar\phi}$ is trained with the logits of Softmax-based MTM and results are presented in Table 5.6. The average scores of the $SMs^{\bigstar\phi}$ indicates distinguishable increase in F1-score compared with the $SMs^{\bigstar\phi}$. This increase in performance illustrates that Softmax-based teacher MTM distills more knowledge in comparison of CRF-based teacher MTM (MTM^{CRF}). The $SM^{\bigstar\phi}(\alpha = 0)$ achieves F1-score for

Datasets	STM^{CRF}	MTM^{CRF}	CRF-based SM$^{\bigstar}$		
			$\alpha = 0$	$\alpha = 0.5$	$\alpha = 1$
AnatEM	86.83	**87.70**	87.66	*87.67*	87.62
BC2GM	**81.82**	*81.68*	80.17	80.23	80.34
BC4CHEMD	**90.43**	89.10	*90.23*	90.10	90.07
BC5CDR	**88.68**	*88.45*	88.31	88.38	88.29
BioNLP09	87.88	**89.08**	88.03	*88.09*	88.03
BioNLP11EPI	83.41	**85.28**	*84.09*	83.94	83.97
BioNLP11ID	86.21	**87.66**	86.84	*87.10*	86.76
BioNLP13CG	83.18	**84.70**	*83.42*	83.39	83.25
BioNLP13GE	76.65	**80.93**	*77.04*	76.88	76.94
BioNLP13PC	87.69	**89.35**	88.17	88.06	*88.29*
CRAFT	**85.11**	*84.49*	84.38	84.41	84.37
ExPTM	73.54	**82.44**	76.02	*76.06*	75.95
JNLPBA	*72.26*	**72.84**	71.41	71.20	71.22
linnaeus	87.86	88.32	88.86	**89.50**	*88.92*
NCBI	84.85	**86.00**	84.97	*85.07*	84.99
Average	83.76	**85.20**	83.97	*84.01*	83.93

Tab. 5.5: Results comparison of the proposed CRF-based SM. $SM^{\bigstar}$ trained with CRF-based teacher MTM.

nine datasets compared with its counterpart STM. While the $SMs^{\bigstar\phi}(\alpha = 0.5$ and $\alpha = 1)$ show an increase in performance for ten datasets against STM.

Figure 5.6 depicts the statistical analyses of the CRF-based SMs. The MTM^{CRF} and STM^{CRF} represent the models with CRF at the output layer unlike other MTM and STM where Softmax is used at the output layer. It can be noticed that CRF-based SMs are statistically worse compared with their corresponding teacher MTM^{CRF}. However, the Softmax-based SMs produce statistically better results compared with their corresponding teacher MTM.

Datasets	STM^{CRF}	MTM	CRF-based SM$^{\star\Phi}$		
			$\alpha = 0$	$\alpha = 0.5$	$\alpha = 1$
AnatEM	86.83	86.78	87.48	**87.59**	*87.49*
BC2GM	**81.82**	79.68	81.26	*81.36*	81.20
BC4CHEMD	**90.43**	86.80	*89.81*	89.71	89.78
BC5CDR	**88.68**	87.49	*88.33*	88.31	88.30
BioNLP09	87.88	88.40	**88.94**	*88.79*	88.75
BioNLP11EPI	83.41	**84.56**	83.92	*84.40*	84.38
BioNLP11ID	86.21	**87.26**	*86.82*	86.66	86.69
BioNLP13CG	83.18	**83.83**	83.12	83.31	*83.36*
BioNLP13GE	76.65	**80.06**	77.89	77.80	*77.97*
BioNLP13PC	87.69	88.17	**88.33**	88.25	*88.28*
CRAFT	**85.11**	81.96	84.10	*84.35*	84.32
ExPTM	73.54	**80.69**	76.08	76.08	*76.38*
JNLPBA	**72.26**	70.40	71.82	*71.89*	71.72
linnaeus	87.86	88.32	*88.94*	**89.67**	88.78
NCBI	84.85	84.50	85.33	*85.37*	**85.54**
Average	83.76	83.93	84.14	**84.50**	*84.20*

Tab. 5.6: Results comparison of the proposed CRF-based SM. $SM^{\star\Phi}$ trained with Softmax-based teacher MTM.

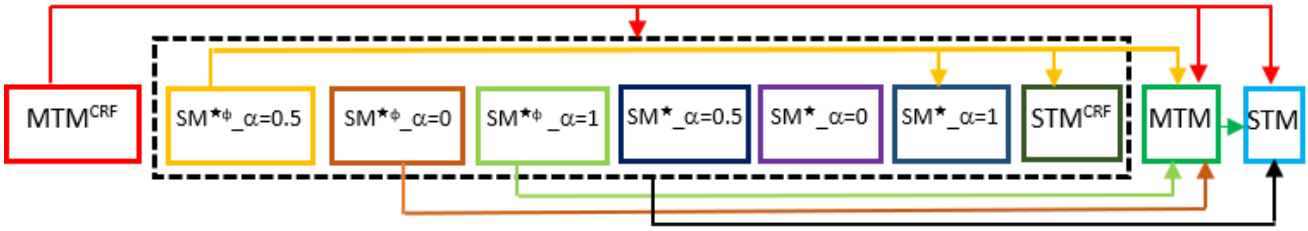


Fig. 5.6: Graphical representation of the Friedman test for Table 5.5 and Table 5.6. The arrows show models are statistically significant with each other. Models are shown according to their ranks starting best model from left to right.

While no SM is able to produce statistically better results against another SM. However, $SMs^{\bigstar^{\phi}}$ trained with Softmax-based MTM are statistically better compared with CRF-based and Softmax-based STMs, whereas $SMs^{\bigstar}$ failed to produce statistically different results against STM^{CRF}.

5.2.5 Knowledge Distillation Using Soft Labels

So far the SMs discussed in the above subsections are trained on the true labels along with the feature representations of the teacher model's intermediate layers. This subsection presents the $SM^{\P}$ that is trained on the soft labels of the teacher model simultaneously. The soft labels are in fact the output probability distribution of the Softmax function. However, they are normalized by a constant temperature hyperparameter τ as discussed in section 2.6.5.The cross entropy loss is calculated for soft labels just as cross entropy is used for true labels. Both losses, cross entropy loss of true labels and cross entropy loss of soft labels, are considered equally. Table 5.7 depicts the results of the $SM^{\P}$ where it can be seen that the $SM^{\P}$ shows a slight performance gain for some datasets compared with the STM. While comparing with the MTM, the $SM^{\P}$ is relatively found better for those datasets, for which STM has shown high F1-score e.g., BC2GM, BC4CHEMD, and CRAFT. This indicates that the learning behaviour of the $SM^{\P}$ resembles more towards STM. By analyzing the results, it can be concluded that logits pass more knowledge to the student model (section 5.2.1) compared to the Soft labels.

The output of the Friedman test is shown in Figure 5.7 where it can be seen that $SM^{\P}$ performs statistically worse than the MTM. The $SM^{\P}$ is also not able to produce significant results compared with the STM.

Datasets	STM	MTM	$SM^{\P}$
AnatEM	86.63	*86.75*	**86.82**
BC2GM	**81.12**	*79.82*	**81.12**
BC4CHEMD	**90.19**	86.71	*90.18*
BC5CDR	*88.06*	87.38	**88.20**
BioNLP09	87.49	**88.35**	*87.57*
BioNLP11EPI	82.66	**84.49**	*83.03*
BioNLP11ID	*85.65*	**87.25**	85.58
BioNLP13CG	82.11	**83.91**	*82.24*
BioNLP13GE	*75.51*	**79.67**	75.00
BioNLP13PC	87.03	**88.37**	*87.09*
CRAFT	*84.32*	82.14	**84.38**
ExPTM	*73.28*	**80.80**	73.05
JNLPBA	*70.82*	70.37	**71.00**
linnaeus	*87.66*	**88.34**	87.63
NCBI	*84.15*	**84.74**	84.02
Average	83.11	**83.94**	*83.13*

Tab. 5.7: Results comparison of the $SM^{\P}$ trained with soft labels of teacher MTM.

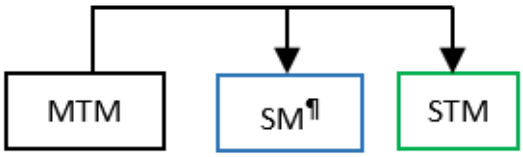


Fig. 5.7: Graphical representation of the Friedman test for Table 5.7. The arrows show models that are statistically significant with each other. Models are shown according to their ranks starting best model from left to right.

6. CONCLUSION

This thesis has explored different knowledge transfer techniques to leverage the performance of the BioNER task. The first approach presented in this research is based on the multi-task learning (MTL) approach. Jointly training various models helps multiple tasks to transfer their knowledge implicitly using a shared layer(s). MTL, therefore, increases the size of the data available to the multi-task model (MTM). MTL also allows the model to learn those features, which can be more challenging to learn independently from any specific task. In this thesis, two different MTMs are proposed: a multi-task model with a convolutional neural network (MTM-CNN) and a multi-task model with character and word input (MTM-CW). Based on the results presented in this thesis, we find that MTM indeed leverages the performance of the BioNER task over a single task model (STM).

The proposed MTM-CNN is trained with the same BioNER task and with dissimilar auxiliary tasks: GENIA-POS tagging and CoNLL chunking (section 3.3.3). The results from this work showed that MTM learns complex features more proficiently when trained with dissimilar auxiliary tasks. The results also showed that the main task comprehends complex features from auxiliary tasks if they are trained at the lower layers (innermost) of MTM. This suggests that some tasks learn from other tasks in an expected order in the MTM. In this way, some tasks may provide supervision to other tasks in a more efficient way.

The MTM-CW propagates the input embedding information along with the outputs of different shared layers to the subsequent layers. This allows successive layers to learn the complex structure from the encoded represen-

tation of the previous layers. MTM-CW yields high F1-score and Friedman output ranks, compared with the MTM-CNN and MTM-CW$^{w/o}$ (the MTM-CW without prior encoded representation of the preceding lower layers).

We also modified the MTM-CNN and MTM-CW at the output layer, where Softmax is used instead of the conditional random field (CRF). The CRF-based MTMs achieve better results and are statistically superior than the Softmax-based MTMs (section 3.3.2, section 3.5.1). The performance degradation with Softmax could be due to the ambiguous entities present in the data. The Softmax function produces an output probability distribution for a specific word and ignores the information of the neighbour words.

Another method used in the thesis is based on the transfer learning technique. The transfer learning is applied on the pre-trained MTM. The MTM is trained for different epochs as an auxiliary task which are then further fine-tuned for a specific task. The objective is to produce a generalized base model which is then specialized for a specific dataset. The results show a performance gain compared to the MTM-CNN, MTM-CW, and other state-of-the-art approaches including STM. The experiments showed that training the base model on dominant data could lead to a performance drop when fine-tuned on similar data, speculating this causes the target model to be stuck in a local minimal. The statistical analysis depicts that the proposed transfer learning approach produces statistically significant results against MTM-CNN, MTM-CW, and STM.

The last method presented in this thesis is based on the knowledge distillation approach. The MTM has the ability to generalize well by learning common features among various tasks which has been seen in the results reported in the thesis. In order to transfer the generalizations of the MTM, the proposed knowledge distillation method utilizes MTM as the teacher model. The Deep learning model learns from generic level features to abstract level features, layers by layers. The proposed approach, therefore, performs knowledge distillation from different layers of the MTM that include

shared BiLSTM, task specific BiLSTM, and a hidden layer.

Additionally, an ensemble method is also implemented where logits of the different MTMs are averaged to train the student model. In another ensemble approach, the logits of the Softmax-based MTM and CRF-based MTM are averaged to teach the student model. The distillation and student losses are controlled by their tunable parameters. The results show that the values of these hyperparameters could depend on the structure and the size of both teacher and student models. However, a performance gain is noted for student models when both, true labels loss and distillation loss, are considered equally. The results from this thesis also revealed that distilling knowledge from the Softmax-based MTM is more favorable for knowledge distillation compared with the CRF-based MTM.

For future work, the MTL approach will be extended by combing different auxiliary tasks with BioNER. This can include biomedical relation extraction, biomedical question answering tasks, etc. The underlying relationship between auxiliary tasks and main tasks will be explored to understand the selection of auxiliary tasks for the main task and this can also help to remove the training data that causes a bottleneck during MTM training. The results of section 3.3.3 have showed that auxiliary tasks trained at the innermost layer consistently produced better results. In future work, this hierarchical relationship between main and auxiliary tasks will be further investigated with more auxiliary tasks trained at different levels of layers.

The knowledge distillation will be extended for semi-supervised learning where MTM will be used to produce soft labels for the large amount of unlabeled data. These soft targets will then be used to train a student model. Additionally, the training of the student model can be combined with noisy data and soft labels to investigate its effect on the performance of the student model. The knowledge distillation approach will also be extended for model compression, where the complicated model will be utilized to teach a simpler student model.

BIBLIOGRAPHY

[1] Gamal Crichton, Sampo Pyysalo, Billy Chiu, and Anna Korhonen. A neural network multi-task learning approach to biomedical named entity recognition. *BMC bioinformatics*, 18(1):368, 2017.

[2] Mingbin Xu, Hui Jiang, and Sedtawut Watcharawittayakul. A local detection approach for named entity recognition and mention detection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1237–1247, 2017.

[3] David Campos, Sérgio Matos, and José Luís Oliveira. Biomedical named entity recognition: a survey of machine-learning tools. *Theory and Applications for Advanced Text Mining*, pages 175–195, 2012.

[4] Jie Zhang, Dan Shen, Guodong Zhou, Jian Su, and Chew-Lim Tan. Enhancing hmm-based biomedical named entity recognition by studying special phenomena. *Journal of biomedical informatics*, 37(6):411–422, 2004.

[5] Burr Settles. Biomedical named entity recognition using conditional random fields and rich feature sets. In *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications (NLPBA/BioNLP)*, pages 107–110, 2004.

[6] Beatrice Alex, Barry Haddow, and Claire Grover. Recognising nested

named entities in biomedical text. In *Biological, translational, and clinical language processing*, pages 65–72, 2007.

[7] Hye-Jeong Song, Byeong-Cheol Jo, Chan-Young Park, Jong-Dae Kim, and Yu-Seop Kim. Comparison of named entity recognition methodologies in biomedical documents. *Biomedical engineering online*, 17(2):158, 2018.

[8] Yi-Feng Lin, Tzong-Han Tsai, Wen-Chi Chou, Kuen-Pin Wu, Ting-Yi Sung, and Wen-Lian Hsu. A maximum entropy approach to biomedical named entity recognition. In *Proceedings of the 4th International Conference on Data Mining in Bioinformatics*, pages 56–61. Citeseer, 2004.

[9] Shigeaki Sakurai. *Theory and Applications for Advanced Text Mining*. BoD–Books on Demand, 2012.

[10] Dan Claudiu Ciresan, Ueli Meier, Luca Maria Gambardella, and Jürgen Schmidhuber. Convolutional neural network committees for handwritten character classification. In *2011 International Conference on Document Analysis and Recognition, ICDAR 2011, Beijing, China, September 18-21, 2011*, pages 1135–1139. IEEE Computer Society, 2011.

[11] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3320–3328, 2014.

[12] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531, 2015.

[13] Michael R Sheldon, Michael J Fillyaw, and W Douglas Thompson. The use and interpretation of the Friedman test in the analysis of ordinal-scale data in repeated measures designs. *Physiotherapy Research International*, 1(4):221–228, 1996.

[14] Richard Tzong-Han Tsai, Shih-Hung Wu, Wen-Chi Chou, Yu-Chun Lin, Ding He, Jieh Hsiang, Ting-Yi Sung, and Wen-Lian Hsu. Various criteria in the evaluation of biomedical named entity recognition. *BMC Bioinform.*, 7:92, 2006.

[15] Erik F. Tjong Kim Sang and Fien De Meulder. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In Walter Daelemans and Miles Osborne, editors, *Proceedings of the Seventh Conference on Natural Language Learning, CoNLL 2003, Held in cooperation with HLT-NAACL 2003, Edmonton, Canada, May 31 - June 1, 2003*, pages 142–147. ACL, 2003.

[16] Sampo Pyysalo and Sophia Ananiadou. Anatomical entity mention recognition at literature scale. *Bioinformatics*, 30(6):868–875, 2014.

[17] Sumam Francis, Jordy Van Landeghem, and Marie-Francine Moens. Transfer learning for named entity recognition in financial and biomedical documents. *Information*, 10(8):248, 2019.

[18] Martin Krallinger, Florian Leitner, Obdulia Rabal, Miguel Vazquez, Julen Oyarzabal, and Alfonso Valencia. Chemdner: The drugs and chemical names extraction challenge. *Journal of cheminformatics*, 7(S1):S1, 2015.

[19] Chih-Hsuan Wei, Yifan Peng, Robert Leaman, Allan Peter Davis, Carolyn J Mattingly, Jiao Li, Thomas C Wiegers, and Zhiyong Lu. Overview of the biocreative v chemical disease relation (cdr) task. In *Proceedings of the fifth BioCreative challenge evaluation workshop*, volume 14, 2015.

[20] Jin-Dong Kim, Tomoko Ohta, Sampo Pyysalo, Yoshinobu Kano, and Jun'ichi Tsujii. Overview of bionlp'09 shared task on event extraction. In *Proceedings of the BioNLP 2009 workshop companion volume for shared task*, pages 1–9, 2009.

[21] Tomoko Ohta, Sampo Pyysalo, and Jun'ichi Tsujii. Overview of the epigenetics and post-translational modifications (epi) task of bionlp shared task 2011. In *Proceedings of BioNLP Shared Task 2011 Workshop*, pages 16–25, 2011.

[22] Sampo Pyysalo, Tomoko Ohta, Makoto Miwa, and Jun'ichi Tsujii. Towards exhaustive protein modification event extraction. In *Proceedings of BioNLP 2011 Workshop*, pages 114–123, 2011.

[23] Sampo Pyysalo, Tomoko Ohta, Rafal Rak, Dan Sullivan, Chunhong Mao, Chunxia Wang, Bruno Sobral, Jun'ichi Tsujii, and Sophia Ananiadou. Overview of the infectious diseases (id) task of bionlp shared task 2011. In *Proceedings of BioNLP Shared Task 2011 Workshop*, pages 26–35, 2011.

[24] Claire Nédellec, Robert Bossy, Jin-Dong Kim, Jung-Jae Kim, Tomoko Ohta, Sampo Pyysalo, and Pierre Zweigenbaum. Overview of bionlp shared task 2013. In *Proceedings of the BioNLP shared task 2013 workshop*, pages 1–7, 2013.

[25] Sampo Pyysalo, Tomoko Ohta, and Sophia Ananiadou. Overview of the cancer genetics (cg) task of bionlp shared task 2013. In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pages 58–66, 2013.

[26] Jin-Dong Kim, Jung-jae Kim, Xu Han, and Dietrich Rebholz-Schuhmann. Extending the evaluation of genia event task toward knowledge base construction and comparison to gene regulation ontology task. *BMC bioinformatics*, 16(S10):S3, 2015.

[27] Jin-Dong Kim, Yue Wang, and Yamamoto Yasunori. The genia event extraction shared task, 2013 edition-overview. In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pages 8–15, 2013.

[28] Tomoko Ohta, Sampo Pyysalo, Rafal Rak, Andrew Rowley, Hong-Woo Chun, Sung-Jae Jung, Sung-Pil Choi, Sophia Ananiadou, and Jun'ichi Tsujii. Overview of the pathway curation (pc) task of bionlp shared task 2013. In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pages 67–75, 2013.

[29] Abdur Rahman MA Basher, Alexander S Purdy, and Inanç Birol. Event extraction from biomedical literature. *bioRxiv*, page 034397, 2015.

[30] Huaiyu Mi and Paul Thomas. Panther pathway: an ontology-based pathway database coupled with data analysis tools. In *Protein Networks and Pathway Analysis*, pages 123–140. Springer, 2009.

[31] Michael Bada, Miriam Eckert, Donald Evans, Kristin Garcia, Krista Shipley, Dmitry Sitnikov, William A Baumgartner, K Bretonnel Cohen, Karin Verspoor, Judith A Blake, et al. Concept annotation in the craft corpus. *BMC bioinformatics*, 13(1):161, 2012.

[32] Jin-Dong Kim, Tomoko Ohta, Yoshimasa Tsuruoka, Yuka Tateisi, and Nigel Collier. Introduction to the bio-entity recognition task at jnlpba. In *Proceedings of the international joint workshop on natural language processing in biomedicine and its applications*, pages 70–75. Citeseer, 2004.

[33] Nhung TH Nguyen, Roselyn S Gabud, and Sophia Ananiadou. Copious: A gold standard corpus of named entities towards extracting species occurrence from biodiversity literature. *Biodiversity data journal*, (7), 2019.

[34] Rezarta Islamaj Doğan, Robert Leaman, and Zhiyong Lu. Ncbi disease corpus: a resource for disease name recognition and concept normalization. *Journal of biomedical informatics*, 47:1–10, 2014.

[35] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 3111–3119, 2013.

[36] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Trans. Assoc. Comput. Linguistics*, 5:135–146, 2017.

[37] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In Carla E. Brodley and Andrea Pohoreckyj Danyluk, editors, *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001), Williams College, Williamstown, MA, USA, June 28 - July 1, 2001*, pages 282–289. Morgan Kaufmann, 2001.

[38] Lynette Hirschman, Alexander A Morgan, and Alexander S Yeh. Rutabaga by any other name: extracting biological names. *Journal of Biomedical Informatics*, 35(4):247–259, 2002.

[39] Yoshimasa Tsuruoka and Jun'ichi Tsujii. Probabilistic term variant generator for biomedical terms. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 167–173, 2003.

[40] Wen-Juan Hou and Hsin-Hsi Chen. Enhancing performance of protein and gene name recognizers with filtering and integration strategies. *Journal of Biomedical Informatics*, 37(6):448–460, 2004.

[41] Meenakshi Narayanaswamy, KE Ravikumar, and K Vijay-Shanker. A biological named entity recognizer. In *Biocomputing 2003*, pages 427–438. World Scientific, 2002.

[42] James Thomas, David Milward, Christos Ouzounis, Stephen Pulman, and Mark Carroll. Automatic extraction of protein interactions from scientific abstracts. In *Biocomputing 2000*, pages 541–552. World Scientific, 1999.

[43] Sophia Ananiadou. A methodology for automatic term recognition. In *COLING 1994 Volume 2: The 15th International Conference on Computational Linguistics*, 1994.

[44] Nigel Collier, Chikashi Nobata, and Jun'ichi Tsujii. Extracting the names of genes and gene products with a hidden markov model. In *COLING 2000 Volume 1: The 18th International Conference on Computational Linguistics*, 2000.

[45] Takaki Makino, Yoshihiro Ohta, Jun'ichi Tsujii, et al. Tuning support vector machines for biomedical named entity recognition. In *Proceedings of the ACL-02 workshop on Natural language processing in the biomedical domain*, pages 1–8, 2002.

[46] Dan Shen, Jie Zhang, Guodong Zhou, Jian Su, and Chew Lim Tan. Effective adaptation of hidden markov model-based named entity recognizer for biomedical domain. In *Proceedings of the ACL 2003 workshop on Natural language processing in biomedicine*, pages 49–56, 2003.

[47] Ki-Joong Lee, Young-Sook Hwang, Seonho Kim, and Hae-Chang Rim.

Biomedical named entity recognition using two-phase model based on svms. *Journal of Biomedical Informatics*, 37(6):436–447, 2004.

[48] GuoDong Zhou and Jian Su. Exploring deep knowledge resources in biomedical name recognition. In *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications (NLPBA/BioNLP)*, pages 99–102, 2004.

[49] Jenny Rose Finkel, Shipra Dingare, Huy Nguyen, Malvina Nissim, Christopher D. Manning, and Gail Sinclair. Exploiting context for biomedical entity recognition: From syntax to the web. In Nigel Collier, Patrick Ruch, and Adeline Nazarenko, editors, *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications, NLPBA/BioNLP 2004, Geneva, Switzerland, August 28-29, 2004*, 2004.

[50] Maryam Habibi, Leon Weber, Mariana Neves, David Luis Wiegandt, and Ulf Leser. Deep learning with word embeddings improves biomedical named entity recognition. *Bioinformatics*, 33(14):i37–i48, 2017.

[51] Qile Zhu, Xiaolin Li, Ana Conesa, and Cécile Pereira. Gram-cnn: a deep learning approach with local context for named entity recognition in biomedical text. *Bioinformatics*, 34(9):1547–1554, 2017.

[52] Wonjin Yoon, Chan Ho So, Jinhyuk Lee, and Jaewoo Kang. Collabonet: collaboration of deep neural networks for biomedical named entity recognition. *BMC bioinformatics*, 20(10):249, 2019.

[53] Xuan Wang, Yu Zhang, Xiang Ren, Yuhao Zhang, Marinka Zitnik, Jingbo Shang, Curtis Langlotz, and Jiawei Han. Cross-type biomedical named entity recognition with deep multi-task learning. *Bioinformatics*, 35(10):1745–1752, 2019.

[54] Xi Wang, Jiagao Lyu, Li Dong, and Ke Xu. Multitask learning for biomedical named entity recognition with cross-sharing structure. *BMC bioinformatics*, 20(1):427, 2019.

[55] John M Giorgi and Gary D Bader. Transfer learning for biomedical named entity recognition with neural networks. *Bioinformatics*, 34(23):4087–4094, 2018.

[56] S. Albawi, T. A. Mohammed, and S. Al-Zawi. Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)*, pages 1–6, 2017.

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

[58] Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

[59] Tom M Mitchell. *The need for biases in learning generalizations*.

[60] Trapit Bansal, David Belanger, and Andrew McCallum. Ask the gru: Multi-task learning for deep text recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 107–114. ACM, 2016.

[61] Jonathan Baxter. A model of inductive bias learning. *J. Artif. Intell. Res.*, 12:149–198, 2000.

[62] Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. Multi-task sequence to sequence learning. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[63] Lin Wang and Kuk-Jin Yoon. Knowledge distillation and student-teacher learning for visual intelligence: A review and new outlooks. *CoRR*, abs/2004.05937, 2020.

[64] Asit K. Mishra and Debbie Marr. Apprentice: Using knowledge distillation techniques to improve low-precision network accuracy. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[65] Przemyslaw Kazienko, Edwin Lughofer, and Bogdan Trawinski. Hybrid and ensemble methods in machine learning. *J. Univers. Comput. Sci.*, 19(4):457–461, 2013.

[66] Lars Kai Hansen and Peter Salamon. Neural network ensembles. *IEEE Trans. Pattern Anal. Mach. Intell.*, 12(10):993–1001, 1990.

[67] Guodong Zhou, Dan Shen, Jie Zhang, Jian Su, and Soon-Heng Tan. Recognition of protein/gene names from text using an ensemble of classifiers. *BMC Bioinform.*, 6(S-1), 2005.

[68] Manabu Torii, Zhang-Zhi Hu, Cathy H. Wu, and Hongfang Liu. Research paper: Biotagger-gm: A gene/protein name recognition system. *J. Am. Medical Informatics Assoc.*, 16(2):247–255, 2009.

[69] Son Doan, Nigel Collier, Hua Xu, Pham Duy, and Tu Minh Phuong. Recognition of medication information from discharge summaries using ensembles of classifiers. *BMC Medical Informatics Decis. Mak.*, 12:36, 2012.

[70] Md Faisal Mahbub Chowdhury and Alberto Lavelli. Disease mention recognition with specific features. In *Proceedings of the 2010 Workshop on Biomedical Natural Language Processing*, pages 83–90, 2010.

[71] Ivano Lauriola, Sella Riccardo, Aiolli Fabio, Alberto Lavelli, and Fabio Rinaldi. Learning representations for biomedical named entity recognition. In *2nd workshop on Natural Language for Artificial Intelligence (NL4AI 2018)*, pages 83–94, 2018.

[72] Ross B. Girshick. Fast R-CNN. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 1440–1448. IEEE Computer Society, 2015.

[73] Li Deng, Geoffrey E. Hinton, and Brian Kingsbury. New types of deep neural network learning for speech recognition and related applications: an overview. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2013, Vancouver, BC, Canada, May 26-31, 2013*, pages 8599–8603. IEEE, 2013.

[74] Bharath Ramsundar, Steven M. Kearnes, Patrick Riley, Dale Webster, David E. Konerding, and Vijay S. Pande. Massively multitask networks for drug discovery. *CoRR*, abs/1502.02072, 2015.

[75] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: deep neural networks with multitask learning. In William W. Cohen, Andrew McCallum, and Sam T. Roweis, editors, *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*, volume 307 of *ACM International Conference Proceeding Series*, pages 160–167. ACM, 2008.

[76] Marcel Bollmann and Anders Søgaard. Improving historical spelling normalization with bi-directional lstms and multi-task learning. *arXiv preprint arXiv:1610.07844*, 2016.

[77] Nanyun Peng and Mark Dredze. Multi-task multi-domain representation learning for sequence tagging. *CoRR, abs/1608.02689*, 2016.

[78] Barbara Plank, Anders Søgaard, and Yoav Goldberg. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. *arXiv preprint arXiv:1604.05529*, 2016.

[79] Zhilin Yang, Ruslan Salakhutdinov, and William Cohen. Multi-task cross-lingual sequence tagging from scratch. *arXiv preprint arXiv:1603.06270*, 2016.

[80] Yuan Zhang and David Weiss. Stack-propagation: Improved representation learning for syntax. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics, 2016.

[81] Richard Johansson. Training parsers on incompatible treebanks. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 127–137, 2013.

[82] Anders Søgaard and Yoav Goldberg. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 2: Short Papers*. The Association for Computer Linguistics, 2016.

[83] Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. A joint many-task model: Growing a neural network for multiple NLP tasks. In Martha Palmer, Rebecca Hwa, and Sebastian Riedel, editors, *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 1923–1933. Association for Computational Linguistics, 2017.

[84] Joan Serrà, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 4555–4564. PMLR, 2018.

[85] Donald W Zimmerman and Bruno D Zumbo. Relative power of the Wilcoxon test, the Friedman test, and repeated-measures ANOVA on ranks. *The Journal of Experimental Education*, 62(1):75–86, 1993.

[86] Fabrice Dugas and Eric Nichols. DeepNNNER: Applying BLSTM-CNNs and extended lexicons to named entity recognition in tweets. In *Proceedings of the 2nd Workshop on Noisy User-generated Text (WNUT)*, pages 178–187, 2016.

[87] Isabel Segura-Bedmar, Víctor Suárez-Paniagua, and Paloma Martínez. Exploring word embedding for drug name recognition. In *Proceedings of the Sixth International Workshop on Health Text Mining and Information Analysis, Louhi@EMNLP 2015, Lisbon, Portugal, September 17, 2015*, pages 64–72, 2015.

[88] Zhiheng Huang, Wei Xu, and Kai Yu. Bidirectional LSTM-CRF models for sequence tagging. *CoRR*, abs/1508.01991, 2015.

[89] Nut Limsopatham and Nigel Collier. Learning orthographic features in bi-directional LSTM for biomedical named entity recognition. In *Proceedings of the Fifth Workshop on Building and Evaluating Resources for Biomedical Text Mining, BioTxtM@COLING 2016, Osaka, Japan, December 12, 2016*, pages 10–19, 2016.

[90] Cıcero dos Santos, Victor Guimaraes, RJ Niterói, and Rio de Janeiro. Boosting named entity recognition with neural character embeddings.

In *Proceedings of NEWS 2015 The Fifth Named Entities Workshop*, page 25, 2015.

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

[92] Jie Yang, Shuailong Liang, and Yue Zhang. Design challenges and misconceptions in neural sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018*, pages 3879–3889, 2018.

[93] Zhilin Yang, Ruslan Salakhutdinov, and William W. Cohen. Multi-task cross-lingual sequence tagging from scratch. *CoRR*, abs/1603.06270, 2016.

[94] Ling Luo, Zhihao Yang, Pei Yang, Yin Zhang, Lei Wang, Hongfei Lin, and Jian Wang. An attention-based bilstm-crf approach to document-level chemical named entity recognition. *Bioinform.*, 34(8):1381–1388, 2018.

[95] Tahir Mehmood, Alfonso Gerevini, Alberto Lavelli, and Ivan Serina. Leveraging multi-task learning for biomedical named entity recognition. In *International Conference of the Italian Association for Artificial Intelligence*. Springer, 2019.

[96] Jinyu Li, Changliang Liu, and Yifan Gong. Layer trajectory LSTM. In *Interspeech 2018, 19th Annual Conference of the International Speech Communication Association, Hyderabad, India, 2-6 September 2018.*, pages 1768–1772, 2018.

[97] Tahir Mehmood, Alfonso Gerevini, Alberto Lavelli, and Ivan Serina. Multi-task learning applied to biomedical named entity recognition task. In *CLiC-it*, 2019.

[98] Sinno Jialin Pan, Ivor W. Tsang, James T. Kwok, and Qiang Yang. Domain adaptation via transfer component analysis. *IEEE Trans. Neural Networks*, 22(2):199–210, 2011.

[99] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019.

[100] Zhenghao Wu, Hao Zheng, Jianming Wang, Weifeng Su, and Jefferson Fong. BNU-HKBU UIC NLP team 2 at SemEval-2019 task 6: Detecting offensive language using BERT model. In Jonathan May, Ekaterina Shutova, Aurélie Herbelot, Xiaodan Zhu, Marianna Apidianaki, and Saif M. Mohammad, editors, *Proceedings of the 13th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2019, Minneapolis, MN, USA, June 6-7, 2019*, pages 551–555. Association for Computational Linguistics, 2019.

[101] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. In Iryna Gurevych and Yusuke Miyao, editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 328–339. Association for Computational Linguistics, 2018.

[102] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding with unsupervised learning. *Technical report, OpenAI*, 2018.

[103] Maxime Oquab, Léon Bottou, Ivan Laptev, and Josef Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014*, pages 1717–1724. IEEE Computer Society, 2014.

[104] Samir Al-Stouhi and Chandan K. Reddy. Transfer learning for class imbalance problems with inadequate data. *Knowl. Inf. Syst.*, 48(1):201–228, 2016.

[105] Jie Yang, Yue Zhang, and Fei Dong. Neural word segmentation with rich pretraining. In Regina Barzilay and Min-Yen Kan, editors, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 839–849. Association for Computational Linguistics, 2017.

[106] Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. Transfer learning for low-resource neural machine translation. In Jian Su, Xavier Carreras, and Kevin Duh, editors, *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 1568–1575. The Association for Computational Linguistics, 2016.

[107] Xuezhe Ma and Eduard H. Hovy. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics, 2016.

[108] Nehemia Sugianto, Dian Tjondronegoro, Golam Sorwar, Prithwi Chakraborty, and Elizabeth Irenne Yuwono. Continuous learning without forgetting for person re-identification. In *2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–8. IEEE, 2019.

[109] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

[110] Tahir Mehmood, Alfonso Emilio Gerevini, Alberto Lavelli, and Ivan Serina. Combining multi-task learning with transfer learning for biomedical named entity recognition. In Matteo Cristani, Carlos Toro, Cecilia Zanni-Merk, Robert J. Howlett, and Lakhmi C. Jain, editors, *Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 24th International Conference KES-2020, (Virtual Conference), 16-18 September 2020*, volume 176 of *Procedia Computer Science*, pages 848–857. Elsevier, 2020.

[111] Heechul Jung, Jeongwoo Ju, Minju Jung, and Junmo Kim. Less-forgetting learning in deep neural networks. *CoRR*, abs/1607.00122, 2016.

[112] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[113] Jie Zhou, Ying Cao, Xuguang Wang, Peng Li, and Wei Xu. Deep recurrent models with fast-forward connections for neural machine translation. *Transactions of the Association for Computational Linguistics*, 4:371–383, 2016.

[114] Yoon Kim and Alexander M Rush. Sequence-level knowledge distillation. In *EMNLP*, 2016.

[115] Tahir Mehmood, Ivan Serina, Alberto Lavelli, and Alfonso Gerevini. Knowledge distillation techniques for biomedical named entity recognition. In Pierpaolo Basile, Valerio Basile, Danilo Croce, and Elena Cabrio, editors, *Proceedings of the 4th Workshop on Natural Language for Artificial Intelligence (NL4AI 2020) co-located with the 19th International Conference of the Italian Association for Artificial Intelligence (AI*IA 2020), (Virtual Conference), November 25th-27th, 2020*, volume 2735 of *CEUR Workshop Proceedings*, pages 141–156. CEUR-WS.org, 2020.

[116] Xinyu Wang, Yong Jiang, Nguyen Bach, Tao Wang, Fei Huang, and Kewei Tu. Structure-level knowledge distillation for multilingual sequence labeling. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 3317–3330. Association for Computational Linguistics, 2020.

[117] Raphael Tang, Yao Lu, Linqing Liu, Lili Mou, Olga Vechtomova, and Jimmy Lin. Distilling task-specific knowledge from BERT into simple neural networks. *CoRR*, abs/1903.12136, 2019.