Chapter 4

Leveraging Contextual Information for Sequential Labeling

4.1 Introduction

A 'word' in a sentence carries linguistic information such as lexical categories (noun, verb, etc.), and grammatical features such as gender, number, person, etc. The fine-grained POS categories provide a linguistic clue (syntactic information) to decide the appropriate role of a word within a sentence or phrase. Apart from that, semantic information is also encoded within a word or sentence. POS categories can be used to disambiguate the multiple answers provided by a morphological analyzer. The morphological analyzer captures semantic information in conventional machine translation systems. Machine translation system translates every word from a source language to a target language. Yin et al. [261] observed that the Proper Noun POS category words had not been translated correctly in European languages. They tried to overcome this problem by jointly modeling neural machine translation and POS tagging using Multi-Task learning. A piece of syntactic information obtained from the POS data leads to better encoding of a source-sentence structure during the generation from the machine translation system [173].

Earlier work on POS tagger for canonical Hindi text achieved considerable results of about 97.10% on Universal Dependency dataset [189], which belongs to a single domain. The performance reduces radically after deploying this existing trained model to different domain-specific data or out-of-domain data. Domain-specific data such as Tourism and Disease has its own distributions and has a minimal amount of annotated dataset, considered as low resources, which also causes an Out-of-vocabulary (OOV) words issue.

OOV is a major problem in low resources text processing, faced while training a model on one domain of a language and trying it to another domain of the same language. This problem can be partly countered by incorporating character level information into the model.

4.2 Contributions of Chapter

The present chapter addresses the question of *how to leverage the attention mechanism along with the handcrafted features to improve the performance of sequence labeling for low resource languages.* Here, we explore integrating attention mechanisms into the deep learning model for low resource languages. For this, we train a monotonic chunk-wise attention-based deep learning model to leverage the surface-level relationships among sentence's tokens, enhancing the performance of sequence labeling tasks in domain adaptation settings. Later on, this model was refined through contrastive learning and results were obtained for single and multi-source domain adaptation. Compared to single-source adaptation, significant improvements have been observed after using multi-source domain adaptation. Since it is multi-source domain adaptation experiments based on the deep learning model, the effect of layer's freezing was observed. The multi-source domain adaptation provides competitive results over the baselines. However, some datasets, such as the Universal Dependency treebank, have morphological features that can also help to capture longer surface-level relationships hence those features (some common features have been used that are easily available in the different language UD data that are Tense, Case, Gender, Number, Person and Lemma) are ensembled into the model. We ensemble the

features into the different layers because the model follows layered architecture, where the disambiguator layer provides the highest scores in terms of accuracy and F_1-score.

4.3 Methodology

I have designed a novel deep learning based architecture, named as Monotonic Chunk-wise attention with CNN-GRU-Softmax (MCCGS). To design the MCCGS model architecture, Ma and Hovy [140] architecture has been followed, in which a character and word information has fused into the deep learning-based model before the core component of the model, LSTM or GRU. This model has been extended by the attention mechanism in the proposed deep learning architecture. Recently, attention mechanisms have been gaining success in speech, image and textual processing. For the textual processing in the neural machine translation system, the attention precisely aligns the source and target words in the pair. This alignment considers the contextual information of input with their non-continuous relationships and assigns weights to themselves for the next word. Here, we leveraged the attention's advantage, i.e., contextual information and non-continuous relationship, in an antagonistic measure for POS tagging. Input encoder, contextual encoder, and disambiguator are the essential core components of the proposed MCCGS architecture, as illustrated in Figure 4.1.

4.3.1 Input Encoder

Let $D = \{(x^j, y^j) \,|1 \leq j \leq N\}$ is a labeled sentence. Here, D is a single training sentence which belongs to the training data $X \in \{D_1, D_2, \ldots, D_m\}$. The x^j denotes the word and y^j denotes the corresponding POS label of sentence D. The POS label y_j belongs to the q labels, which are represented as $y_j \in \{y_1, y_2, \ldots, y_q\}$.

In this component, MCCGS takes the given input sentence D to obtain the vector representation of each word x^j in terms of learnable word level embedding and character level embedding. The learnable word embedding captures syntax and semantic information of

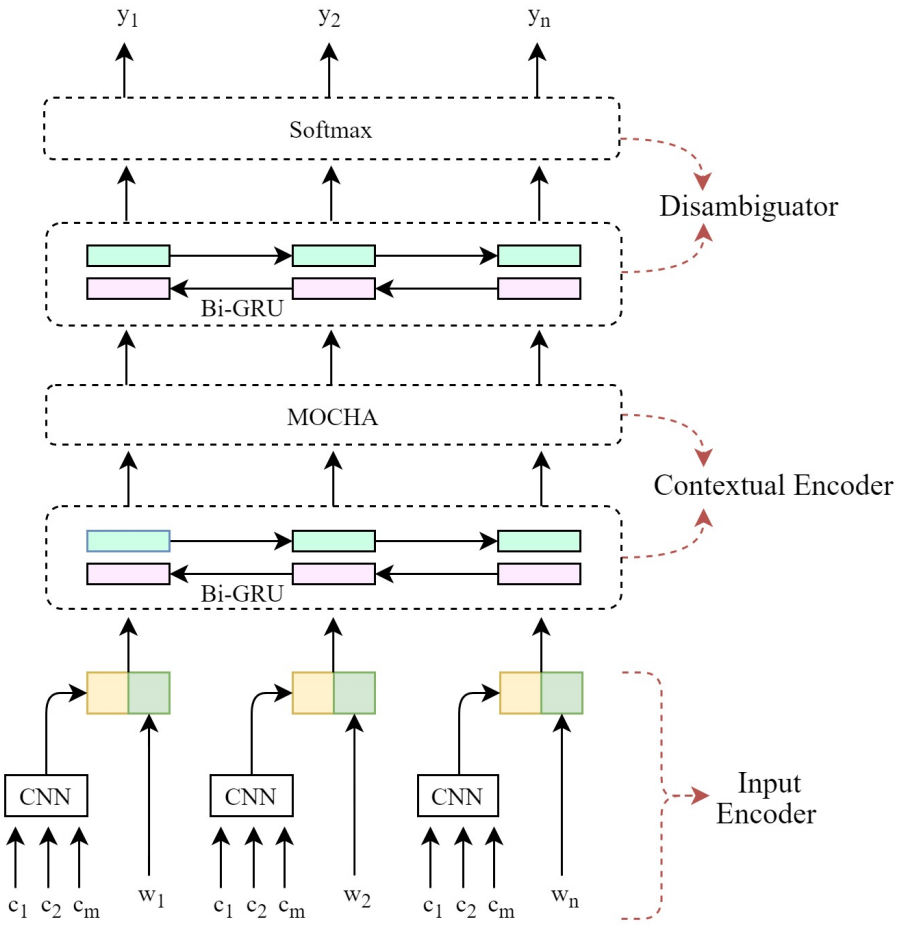

FIGURE 4.1: Overview of the MCCGS architecture with its components

the word which is enhanced through the morphological information from character level embedding.

The learnable word embedding has been obtained from the one-hot vector representation at the size of a unique word vocabulary. All the sentences in X should have length N. If the length of D is bigger than N, the antecedent word will be removed and padding will be applied if the attribute is smaller than N.

This representation and random vector W_x of embedding size, which is trainable, exhibits latent vectors. Such exhibited latent vectors for all words $x^{1:N}$ in a sentence D, attained after passing to a fully connected layer with deactivated bias have been considered word embedding $v^{1:N}$.

$$v^{1:N} = W_x.x^{1:N} \tag{4.1}$$

For instance, the sentence (Figure 4.4) "Tribal communities said they are seeking new financing" considered as input is a sequence of words. Each unique word represents a number in word encoding, and the number becomes a randomly generated real-valued vector. These randomly generated vectors update themselves over the model learning.

For obtaining the character level word embedding, let a word x^j have $\mathcal{C}$ characters, where $\mathcal{C} = \{c^1, c^2, \ldots, c^i | 1 \leq i \leq k\}$. The *character lookup dictionary* maps character identity with its one-hot vector representation. For each character, c^i in x^j is denoted by a one-hot vector representation, which has an equal size of k, using the padding operation. The padding operation appends the special symbol $< pad >$ at the end of the word that padding has applied on the same level at the sentence. If the length of x^j is greater than k, initial characters $c^{i:k}$ have been considered. Now convolution, the first operation of CNN, has been applied with filter F. Let z be the number of filters used with f kernel sizes. The filters move over the region of x^j and generate a set of features C, referred to as *feature maps* which are $\{r_1, r_2, \ldots, r_l\} \in C$. The l is calculated on the basis of kernel size and word length, $l = k - f + 1$.

$$r_i = \phi(F.c_{i:i+k-1} + b_c) \tag{4.2}$$

Here, F, b_c and ϕ are filters, bias and non-linear functions, respectively. A piece of relevant information from all features has leveraged maximum pooling, another building block of a CNN.

$$C_{\max} = \max\{r_1, r_2, \ldots, r_l\} \tag{4.3}$$

$$C = \{C_{\max_1}, C_{\max_2} \ldots, C_{\max_z}\} \tag{4.4}$$

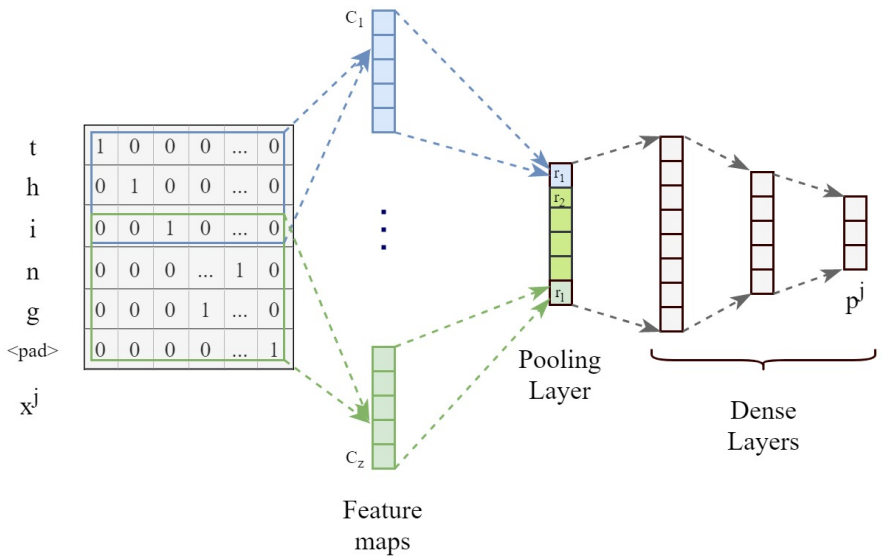

FIGURE 4.2: Character level CNN model architecture to generate word vector

The generated univariant vectors from all features have been concatenated, C and have passed to three stacked fully connected layers. The resultant vector, p^j from the penultimate layer has character level information for the word.

$$p^j = \phi(W_C.C + b_C) \tag{4.5}$$

The W_C and b_C are learning parameters and ϕ is a $ReLU$ non-linear function. This character-level word vector generation process has been illustrated in Figure 4.2. Here, 'thing' is considered as an input word. A padding operation ($< pad >$) has been performed, as the length of the input word is smaller than the desired. All the unique characters of a language dataset were mapped with their index positions. These indexes are character identities that are further used in *character lookup dictionary* to emit a one-hot vector on which CNN operations have been performed and generate character level embeddings. For example, convolution operation generates multiple n-grams, such as thi, hin, ing, ngs, gs$<$ pad $>$ based on the filter size (say 3) that is used in feature maps. Over these generated filters, the pooling operation extracts relevant information passed through the fully connected layer to generate a final vector. This process is iterated over each word which generates $p^{1:N}$.

The word-level embeddings $v^{1:N}$ and character-level embeddings $p^{1:N}$ have been concatenated component-wise to generate the word vector representations $w^{1:N}$. This generated word vector representation leverages morphological, syntactic and semantic information at the word and sentence level.

$$w^{1:N} = [p^{1:N}; v^{1:N}] \tag{4.6}$$

4.3.2 Contextual Encoder

The Context Encoder works in two steps to generate its output. The first step assumes an identical magnitude of words of a sentence to capture sequential information by GRU, a variant of RNN. It allows holding information of a longer timestamp. Although natural language sentences are longer and have dependencies among words, Bi-GRU has been considered.

The Bi-GRU takes the word vector and produces a hidden vector for each direction. The hidden vector h^{i-1} with the word vector w^i decides which information will be forwarded for processing of the succeeding timestamp with a degree of relevance. In each of the timestamps, the forward and backward process generates hidden states, i.e., $\overrightarrow{h}^{1:N}$ and $\overleftarrow{h}^{1:N}$. Forwards $\overrightarrow{h}^{1:N}$ and backwards $\overleftarrow{h}^{1:N}$ hidden vectors have been concatenated according to component-wise, that generates a new hidden vector $h^{1:N}$ as resultant.

$$\overrightarrow{h}^{1:N} = \overrightarrow{GRU}(w^1, w^2, \ldots, w^N) \tag{4.7}$$

$$\overleftarrow{h}^{1:N} = \overleftarrow{GRU}(w^1, w^2, \ldots, w^N) \tag{4.8}$$

$$h^{1:N} = [\overrightarrow{h}^{1:N}; \overleftarrow{h}^{1:N}] \tag{4.9}$$

The second step of the contextual encoder is an attention mechanism. Attention mechanism, which was first introduced for text-based applications [14], focuses on the given input's cogent part to yield a better decision. Numerous tasks based on deep learning, such as image recognition [271], machine translation [271, 242, 63, 14, 103, 39] and text classification [229, 234], have significantly improved their performance after the attention mechanism is deployed. The monotonic chunk-wise attention mechanism explicitly leverages the flexible alignment between input and output, where output is labeled corresponding to the task. The attention is computed over the chunked input sequences.

Let s^{j-1} is disambiguator hidden state at $j-1$, and $h^{1:N} = \{h^1, h^2, \ldots, h^N\}$ is input hidden vectors. The energy e^{ji} calculated as follows:

$$e^{j,i} = MonotonicEnergy(s^{j-1}, h^i) \tag{4.10}$$

Where,

$$MonotonicEnergy(s^{j-1}, h^i) = g \cdot \frac{v^T}{||v||} \tanh(W_s s^{j-1} + W_h h^i + b) + r \tag{4.11}$$

W_s, W_h, r, b, v and g are trainable parameters. The energy scalers for timestamp t_j of the output are obtained from $i = t_{j-1}, t_{j-1}+1, t_{j-1}+2, \ldots, N$. It is passed to the logistic sigmoid function to produce the selection probabilities $p^{j,i}$.

$$p^{j,i} = \sigma(e^{j,i} + \xi), \qquad \xi \sim N(0,1) \tag{4.12}$$

Here, logistic sigmoid with the unit-variance Gaussian noise ξ, constraints selection probabilities $p^{j,i}$ to binary values. The context c^j is generated by the hidden states h^i, which are selected on the $p^{j,i}$. The fixed window length w for hidden states is referred to as a chunk here. Chunk energy is calculated in the same way as monotonic energy but skipping the length normalization of v, g and r. The soft attention over preceding w on hidden states and t_j have been applied for c^j.

$$v = t_j - w + 1 \tag{4.13}$$

$$u^{j,k} = ChunkEnergy(s^{j-1}, h^k); \quad k \in \{v, v+1, v+2, \ldots, t_j\} \tag{4.14}$$

$$c^j = \sum_{k=v}^{t_j} Softmax(u^{j,k}) \cdot h_k \tag{4.15}$$

4.3.3 Disambiguator

Most of the earlier work on sequence labeling depends on long short-term memory and conditional random fields to disambiguate the structured inferences [169]. The hidden state of each timestamp with the associated generated context vector is used to disambiguate the label dependencies using the bi-directional GRU.

$$\overrightarrow{h}^j = \overrightarrow{GRU}(s^{j-1}, c^j) \tag{4.16}$$

$$\overleftarrow{h}^j = \overleftarrow{GRU}(s^{j-1}, c^j) \tag{4.17}$$

$$h^j = [\overrightarrow{h}^j; \overleftarrow{h}^j] \tag{4.18}$$

The h^j is passed to a multi-layer neural network before the softmax layer. This penultimate layer predicts the label sequence after scaling and normalizing the output of a multi-layer neural network.

$$o^j = h^j.W_h + b_h \tag{4.19}$$

$$P(\hat{y}^i|x;\theta) = \frac{\exp(o^j)}{\sum_{i=1}^{l} \exp(o^j)} \tag{4.20}$$

Here, W_h and b_h are the learning parameters. The training objective is to minimize the cross-entropy loss (J) along with the l_2 normalization.

$$J_\theta = -\frac{1}{m}\sum_{d=1}^{m}\sum_{l=1}^{q} y^l \; log(\hat{y}^l) + (1 - y^l) \; log(1 - \hat{y}^l) + \frac{\lambda}{2}||\theta||^2 \tag{4.21}$$

For each sentence d, the model predicts q POS labels in a real-valued vector which is converted into a one-hot vector used to calculate the average difference with the valid POS labels. It is accommodated by cross-entropy since the probability of each label depends on the probability of another label.

4.4 Inclusion of Morphological Features

Deep learning models automatically generate their features according to the given training data and do not depend on handcrafted features as traditional statistical models do. Out of handcrafted features, symbolic features are very common such as lexicon features and affixes, etc. Recent studies have empirically proved that the use of these handcrafted external features improves the model performance [187, 188, 214, 91]. Plank and Klerke [188] have shown the improvement for POS tagging by incorporating the lexicon features at input encoder in cross-lingual settings. However, the deep learning model supports layered architecture, and each layer captures its features. Hence, we have incorporated morphological features at all three components, i.e., at Input Encoder, Contextual Encoder and Disambiguator of the proposed MCCGS model.

Let the $\mathcal{F}$ feature set be available with the training data, which includes $f_1, f_2, f_3, \ldots, f_k$ morphological features. Each feature in a feature set is represented by a vector ($\overrightarrow{f_i}$) to concatenate with each other.

$$\mathcal{F} = \overrightarrow{f_1} \oplus \overrightarrow{f_2} \oplus \overrightarrow{f_3} \oplus \ldots \overrightarrow{f_k} \tag{4.22}$$

The inclusion of morphological features into the MCCGS model is described in the following way. Here, we have described only changes by which MCCGS model architecture extends after performing the feature inclusion since the remaining components of the MCCGS model are the same.

1. The inclusion of this feature set at input encoder of the MCCGS model is expressed by:

$$w^{1:N} = [p^{1:N}; v^{1:N}] \oplus \mathcal{F}^{1:N} \tag{4.23}$$

   Here, $w^{1:N}$ is the resultant word vector which is substituted in the eq. 4.6 of the MCCGS model.

2. While integrating this morphological feature set at the contextual encoder, the final output representation through Bi-GRU (mentioned in eq. 4.9) is changed with the new representation by:

$$h^{1:N} = [\overrightarrow{h}^{1:N}; \overleftarrow{h}^{1:N}] \oplus \mathcal{F}^{1:N} \tag{4.24}$$

3. While including the morphological features at the disambiguator component, the input for the multi-layer neural network has changed. The new input for this layer (eq. 4.18) is represented by:

$$h^{j} = [\overrightarrow{h}^{j}; \overleftarrow{h}^{j}] \oplus \mathcal{F}^{j} \tag{4.25}$$

   Here, the feature set of jth word, $\mathcal{F}^j$ is concatenated with the final hidden representation of the Bi-GRU of the disambiguator layer.

Figure 4.3 depicts the model architecture encompassing the inclusion of morphological features into the Input-Encoder, Contextual Encoder, and Disambiguator components. The inclusion of morphological features is mutually exclusive; therefore, inclusion is performed at a single component. The schema is an extension of Figure 4.1 concerns a feature set $\mathcal{F}$ that consists of the feature vector $\overrightarrow{f_1}, \overrightarrow{f_2}, \overrightarrow{f_3} \ldots \overrightarrow{f_k}$. The feature vectors are concatenated before integrating into each of the components. The output of each concatenator, pink, yellow, and blue, represents Input-Encoder, Contextual Encoder, and Disambiguator components, respectively.

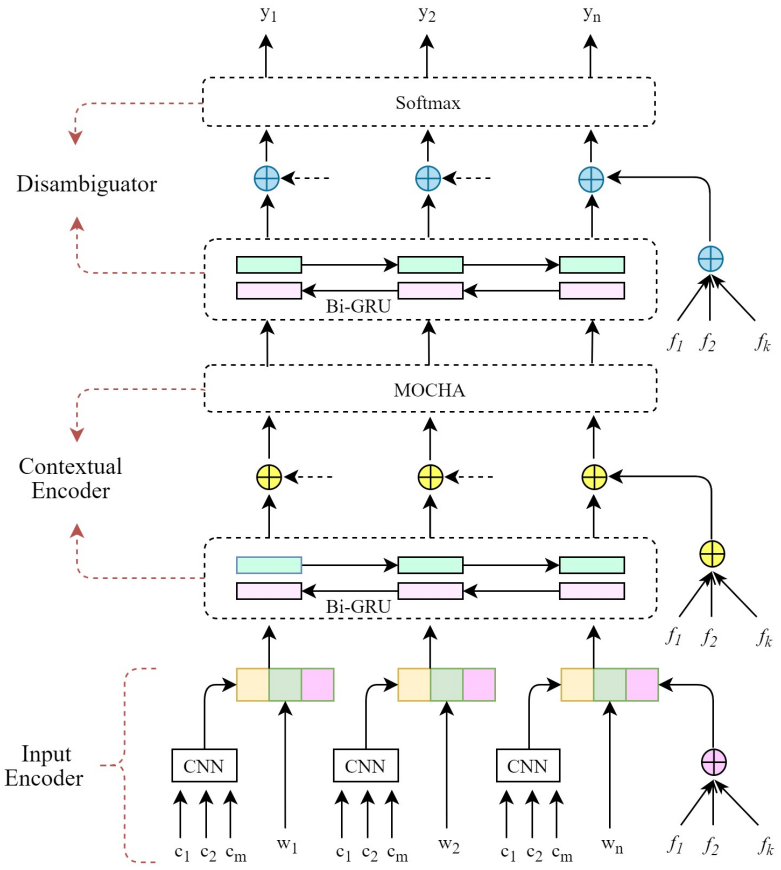

FIGURE 4.3: MCCGS model architecture with symbolic feature inclusion

4.5 Contrastive Training Component

One of the earlier attempts of using contrastive training in NLP applications (text classification) is reported by Miyato et al. [162]. They encoded perturbation into the word embedding to generate a contrastive example. More robustness in the classifier was obtained after adding perturbations (η) to the input examples, which are closer to actual examples. These perturbed examples have the same label as the actual example to which they are close. However, it may be misclassified by the current classifier, which increases the loss, so the current model loss is modified by:

$$\eta = \underset{\eta' : \|\eta'\|_2 \leq \delta}{\arg\max} \; L(\hat{\theta}; S + \eta', Y) \tag{4.26}$$

Here, the normalized perturbation δ allows to construct the contrastive example for each training step:

$$S_{adv} = S + \eta \tag{4.27}$$

The new loss function ($\tilde{L}$) of the contrastive part is based on the classifier having a mixture of actual examples $L(\theta; S, Y)$ and perturbed examples' loss $L(\theta; S_{adv}, Y)$, where individual loss is controlled by the weighting factor γ with 0.5.

$$\tilde{L} = \gamma L(\theta; S, Y) + (1 - \gamma) L(\theta; S_{adv}, Y) \tag{4.28}$$

The pseudo-code of the CMCCGS learning is described in Algorithm 4.1.

Algorithm 4.1 Contrastive Monotonic Chunk-wise attention with CNN-GRU-Softmax (CMCCGS) for POS tagging

Require: N: number of training examples, i.e., sentences
Require: w: chunk size
Require: ϵ: emission score
while s = 1,2,...,N **do**
  Obtain word vectors $x_{i:n}$ by **Eq. (4.1)**
  for $i = 1$ to n **do**
    Obtain character level word vector v_i by **Eq. (4.5)**
  end for
  Obtain word embedding by concatenation according to **Eq. (4.6)**
  Obtain hidden states $h_{1:m}$ on word embedding by **Eq. (4.9)**
  for $i = 1$ to n **do**
    for $j = 1$ to n **do**
      Calculate chunk energy $u_{j,i}$ with w and ϵ by **Eq. (4.14)**
    end for
    Obtain contextual attention vector c_j by **Eq. (4.15)**
    Calculate the hidden vector and probability score to each label by **Eq. (4.18)** and **Eq. (4.20)**
  end for
  Generate contrastive examples by **Eq. (4.27)**
  Calculate loss and label predictions according to **Eq. (4.28)**
  $i \leftarrow i + 1$
end while

4.6 Experiment-I: Feature Integration in MCCGS

With conventional machine learning techniques for POS tagging, there was a heavy reliance on feature engineering such as prefixes, suffixes, contextual words, and language-specific features, i.e., capitalization and creating several handcrafted features. For example, Conditional Random Fields (CRF) and Maximum Entropy (ME) work on symbolic features that include lexicon, affixes and other morphological information to improve POS tagging performance. The availability of lexicons and basic morphological information is ingenious, while dependency information is unfortunately rarely available due to the profound linguistic knowledge required for annotation. If available, these features are directly incorporated into the machine learning algorithms. However, deep learning includes these features as dense representations in the model. Dense feature representation is an adequate representation that learns itself from the provided value of the feature.

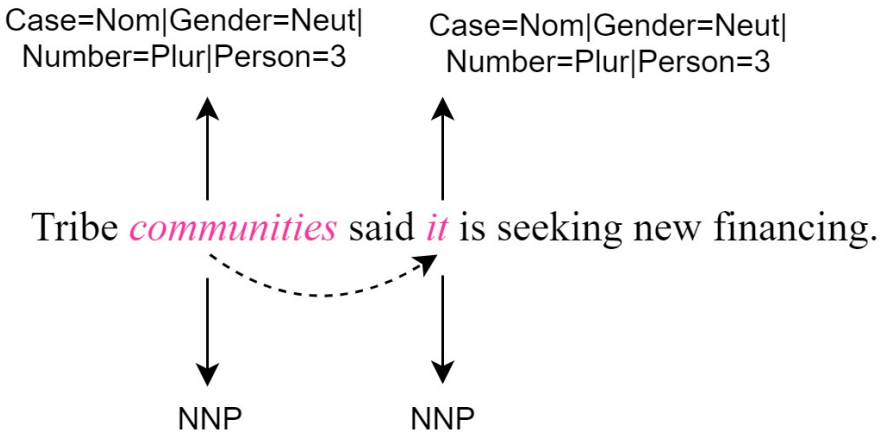

FIGURE 4.4: Impact of morphological features and longer contextual dependencies

Even integrating these dense features in deep learning models cannot effectively address the long-range non-continuous relation dependencies. The non-continuous dependency defined by word depends on its semantic information and depends upon the information of its contextual neighbor; hence, it plays a significant role in POS tagging. This phenomenon has been explained in Figure 4.4, where the word "communities" is dependent upon the word "it", as a result it is tagged with the "NNP" (singular proper noun). If the non-continuous dependency is ignored here, the tag could be "NNPS" (plural proper noun) due to suffix information of '-s' or '-es'. Similarly, the morphological information provides

clues about the category of a word viz. the word "communities" consists of the "Nom" (nominative) case and person information "Person" as 3rd. This information is usually exhibited for the noun categories.

Previously proposed approaches are based on either attention-based contextual information [131, 220, 166] or handcrafted features (existing linguistic knowledge) [187, 188, 34] for non-continuous relation modeling but have not been brought together as needed, which improves the POS tagging performance. Apart from this, there is a less systematic study on feature inclusion regarding at which layer such integration improves the performance of POS tagging, as deep learning follows a layered approach.

Liu et al. [134] used all characters of sentences along with word boundaries for generating word embedding through Bi-LSTM, which was further employed with highway network and Bi-LSTM in a parallel fashion to decode label dependencies by CRF. Zhang et al. [267] proposed a multi-channel model based on the Bi-LSTM for obtaining the word and label dependencies and their interaction simultaneously by using the label of the previous word as a context for the current word in the softmax decoder.

The majority of earlier proposed work on neural POS tagging assumed that handcrafted features were antiquated for deep learning-based models and uniquely depended on an end to end training. However, Faruqui et al. [75] in their earlier work combine semantic symbolic features with word embedding. Similarly, Sagot and Alonso [208] use morphological lexicons as additional input, collected from Apertium and Alexina lexicons (language-specific) as n-hot features in the Plank et al. [189] model, which uses Bi-LSTM for input encoder and CRF for label decoder.

The impact on the deep learning model by using handcrafted features provides a significant gain in performance. Recently, the research communities have been trying to make a robust model by improving self feature learning through the attention mechanism on the assumption of learning of the contextual features for POS tagging. The attention mechanism helps to capture the non-continuous relationship among the words of a sentence. Mundotiya et al. [166] have proposed self-attention and monotonic chunk-wise attention-based model and experimented on the Hindi dataset to handle non-continuous relationships

within a sentence and a separate window by using a respective attention mechanism. Similarly, Wei et al. [247] proposed a new model based on the attention mechanism. Here, standard additive self-attention and a position-aware self-attention mechanism have been exploited to implicitly encode positional information at discrete and variable-length of an input sequence, respectively, to provide complementary context information based on Bi-LSTM as global sequence encoder. Shao et al. [220] used self-attention at the word and sentence level to obtain the contextual information on the same global sequence encoder, which was further used for disambiguating the labels of words by the semi-CRF approach.

There is a less systematic study on improving the performance of POS tagging by handcrafted feature inclusion in deep learning-based models. At what level is the feature integration in those models should be incorporated, as it follows a layered approach is a question in itself. So in this experiment, morphological features are incorporated in a novel neural architecture, Monotonic Chunk-wise attention with CNN-GRU-SOFTMAX (MCCGS) (described in the model section 4.3). Later on, a dense representation of the handcrafted features (morphological features) is included rotation-wise in all three components of the MCCGS model (described in the extended model section 4.4), which is empirically evaluated further.

4.6.1 Dataset Description

The proposed model architecture is validated on an annotated dataset of 21 languages that are available in Universal Dependency (UD) Treebank[1] (version 2.7). The pre-defined splitting of the datasets in the training and development is adopted to train and test our model. The statistics of the training and development datasets are mentioned in Table 5.1.

In this experiment, a sentence with its universal part of speech (UPOS) tags and additional features are considered. These additional features are the Tense, Case, Gender, Number, Person and Lemma obtained from the treebank. These additional features are considered as input to make an accurate prediction.

[1]https://universaldependencies.org/

TABLE 4.1: The languages with their statistics, obtained from the UD treebank. Some languages have more than one treebank; hence the related treebank information is mentioned after - in the language name.

Language	**Training Size**	**Development Size**
Hungarian (hu)	910	441
Greek (el)	1662	403
Swedish-LinES (sv)	3176	1032
Danish (da)	4383	564
Hebrew (he)	5421	484
Croatian (hr)	6914	960
Bulgarian (bg)	8907	1115
Portuguese-GSD (pt)	9664	1210
Dutch-Alpino (nl)	12264	718
English-EWT (en)	12543	2002
Italian-IDST (it)	13121	564
Hindi-HDTB (hi)	13304	1659
German-GSD (de)	13814	799
Spanish-GSD (es)	14187	1400
French-GSD (fr)	14449	1476
Finnish-FTB (fi)	14981	1875
Norwegian-Bokmal (no)	15696	2409
Polish-PDB (pl)	17722	2215
Romanian-Nonstandard (ro)	24122	1052
Persian-PerDT (fa)	26196	1456
Czech-PDT (cs)	68495	9270

4.6.2 Experimental Settings

The proposed model architecture has three core components which are Input Encoder, Contextual Encoder and Disambiguator. The Input Encoder holds a token vector with word and character-level information in a latent vector of 100 and 16 dimensions, respectively. The word latent vector is randomly initialized through uniform distribution of $[-0.05, 0.05]$ [115], whereas the character level vector is initialized as a one-hot vector of dimension 30. The CNN has been applied with two convolution layers of size 64, 124 and a fixed window size of 3 as the kernel, followed by maximum pooling. The number of units in the multilayer feed-forward layer is equal to the size of the convolution layer, which is applied over the one-hot vector to obtain the token vector at character-level [211]. The Bi-GRU has 128 hidden vectors that capture contextual sentence information. The monotonic chunk-wise attention carries dependencies among the adjacent words by the chunk size of

10 and emission probability of 0.6. The label side dependencies are captured by Bi-GRU with 128 hidden units in the disambiguator component. We have used an advanced version of the gradient descent learning algorithm with backpropagation through time (BPTT) to optimize the weight for training the model. The goal of BPTT is to update the weights of a neural network to minimize the error compared to some expected output. It is a supervised learning algorithm that allows the network to be corrected concerning already given labels. The Adam optimizer (advanced version of the gradient descent) has been employed to train the model with a 0.01 initial learning rate and 0.007 decay rate. Here, update in learning rate is defined with $\eta_t = \frac{\eta_0}{1+\rho_t}$, where t is referred to the number of completed epochs, η_0 and ρ are the initial learning rate and decay rate, respectively. The batch size and epochs are fixed during the training, i.e., 32 and 40, respectively. The early stoppage [32] with the patience value of 3 is applied to the validation performance to avoid model overfitting. As an additional regularizer, dropout with a value of 0.5 probability, has also been used.

4.6.3 Results and Analysis

To compare the performance of our proposed model, **Acc**uracy, Precision, Recall, F1-score and Matthews correlation coefficient (MCC) metrics have been used.

Table 4.2 presents tagging accuracy for the 21 individual languages on the development data after applying the MCCGS model. In this table, columns represent Precision, Recall, F1, MCC and Accuracy of each of the corresponding 21 languages. To compare the reported results with state-of-the-art techniques such as, without additional features incorporation, Position-aware Self Attention (PSA) mechanism [247] and with incorporating additional features, Type constraints [235], Retrofitting [75] and Distant Supervision from Disparate Sources [187]. Inclusion of the lexical information by exploiting Type Constraints (TC_w) and Retrofitting (Retro) off-the-shelf embeddings is evaluated in the neural tagging literature. Distant Supervision from Disparate Sources (DsDs) [187] is the alternative way of using lexical information. The replication of feature inclusion results is presented in the three columns towards the right side of Table 4.2. For comparison, the tagging accuracy values from Plank and Klerke [188] have been reused. However, the

TABLE 4.2: Obtained results using the MCCGS model

Lang	**Precision**	**Recall**	**F1**	**MCC**	**Acc**	**TC$_w$**	**Retro**	**DsDs**	**PSA**
hu	83.8	79.59	80.02	75.97	79.59	77.5	75.5	76.2	86.20
nl	93.68	91.44	91.64	91.95	91.44	89.2	86.6	89.6	91.85
pt	91.35	89.95	90.20	89.53	89.95	92.2	88.6	93.1	91.97
es	90.73	89.66	88.99	89.02	89.66	88.9	88.9	91.7	64.58
hi	94.61	94.57	94.47	94.11	94.57	63.9	63.0	66.2	94.85
it	88.57	89.44	88.73	88.26	89.44	91.8	90.0	93.7	88.88
da	91.00	90.55	90.38	90.08	90.55	89.3	88.2	90.1	89.08
fi	93.89	93.13	93.14	92.86	93.13	81.4	79.2	83.1	92.43
el	86.68	87.00	85.77	85.47	87.00	86.1	79.3	79.2	89.55
bg	89.20	90.04	88.93	89.42	90.04	89.9	87.1	91.0	92.46
cs	97.23	95.53	95.75	95.84	95.53	87.5	84.9	87.4	95.14
he	85.12	82.53	82.75	84.67	82.53	75.9	71.7	76.8	88.74
no	96.12	95.55	95.40	93.88	95.55	91.1	88.8	91.4	94.66
fa	95.53	94.90	95.11	94.88	94.90	43.8	44.1	43.6	95.87
sv	92.53	92.22	92.06	92.24	92.22	89.2	87.0	89.8	93.09
en	94.58	94.74	94.36	88.51	94.74	87.6	82.5	87.3	94.02
ro	88.74	86.07	85.47	85.71	86.07	84.2	80.2	86.0	88.02
hr	91.33	89.89	89.60	88.98	89.89	85.2	83.0	85.9	90.50
fr	93.75	93.20	92.52	92.61	93.20	90.0	89.9	91.3	89.39
de	88.41	87.20	86.97	85.43	87.20	87.1	84.7	87.5	90.20
pl	93.87	92.77	92.01	92.29	92.77	84.9	83.9	85.4	92.35

PSA results have been obtained from the entire model training from scratch. In 7 out of 21 languages, the MCCGS model performs better. MCCGS model is better performing model than the retrofitting model in all 21 languages. It also gives better accuracy than the TC$_w$ except for two languages (Portuguese and Italian). DsDs shows better accuracy than all three models (MCCGS, TC$_w$, and Retro) for five languages, Portuguese, Spanish, Italian, Bulgarian and German, which is surprising. On the other hand, the PSA model reports the highest accuracy on twelve languages.

4.6.3.1 Feature inclusion

UD treebank has richness in the lexical and grammatical features of the words since it provides the facility to add language-specific features. Here, we have used those prevalent features which are available in the experimental dataset for all languages, such as Gender, Number, Person, Case, Lemma and Tense. All those features have been concatenated

TABLE 4.3: Obtained scores after inclusion of morphological features in the MCCGS model. Here, IE, CE and D stand for MCCGS-IE, MCCGS-CE and MCCGS-D models respectively.

	Precision			**Recall**			**F-score**			**MCC**			**Accuracy**		
Lang	**IE**	**CE**	**D**	**IE**	**CE**	**D**	**IE**	**CE**	**D**	**IE**	**CE**	**D**	**IE**	**CE**	**D**
hu	81.07	82.89	82.79	81.33	80.00	82.61	79.93	79.95	81.74	77.97	75.84	82.95	81.33	80.00	82.61
nl	92.99	92.37	93.87	93.31	92.18	92.84	92.97	91.63	92.71	91.20	89.48	89.95	93.31	92.18	92.84
pt	94.46	91.25	94.17	94.59	89.81	94.24	94.48	89.99	94.17	94.08	89.74	93.93	94.59	89.81	94.24
es	91.21	91.36	91.12	91.44	90.76	91.39	91.17	90.17	91.08	91.68	91.15	90.71	91.44	90.76	91.39
hi	93.25	94.21	94.81	93.09	93.91	94.77	92.99	93.92	94.63	92.65	93.60	94.42	93.09	93.91	94.77
it	90.23	90.69	92.01	90.91	90.73	92.21	90.33	90.28	91.67	89.88	90.01	91.56	90.91	90.73	92.21
da	91.10	88.76	91.48	91.47	89.24	91.24	91.10	88.60	90.53	91.08	88.29	90.59	91.47	89.24	91.24
fi	93.10	90.78	93.54	93.01	90.92	93.75	92.43	90.51	93.00	92.24	90.11	93.14	93.01	90.92	93.75
el	86.95	90.86	89.63	87.48	87.75	89.53	86.80	87.84	88.34	86.28	87.90	88.35	87.48	87.75	89.53
bg	89.63	86.67	91.85	89.40	88.25	92.55	88.72	86.77	91.72	91.50	85.04	91.70	89.40	88.25	92.55
cs	96.31	96.79	95.76	95.73	96.12	95.80	95.87	96.21	95.57	95.76	96.14	95.49	95.73	96.12	95.80
he	84.38	87.62	87.80	84.68	87.77	87.91	83.65	87.01	87.21	84.84	79.63	86.77	84.68	87.77	87.91
no	96.34	96.54	96.45	95.57	96.25	96.28	95.69	96.30	96.09	95.65	95.95	95.76	95.57	96.25	96.28
fa	95.83	95.83	95.79	95.63	95.73	95.99	95.59	95.60	95.71	95.42	95.42	95.58	95.63	95.73	95.99
sv	93.03	93.79	94.74	92.82	91.04	94.76	92.73	91.44	94.50	92.12	91.22	94.32	92.82	91.04	94.76
en	93.33	93.35	93.95	93.50	93.31	93.74	93.28	93.06	93.71	89.70	87.86	91.82	93.50	93.31	93.74
ro	87.41	88.29	90.39	87.85	86.63	91.16	87.28	85.98	90.31	88.86	88.60	90.10	87.85	86.63	91.16
hr	89.86	91.50	90.55	89.30	89.75	90.41	89.24	89.71	90.00	86.18	89.36	89.72	89.3	89.75	90.41
fr	94.74	94.97	92.77	94.72	94.33	92.97	94.53	94.44	92.53	94.33	94.24	92.35	94.72	94.33	92.97
de	91.01	91.85	91.45	90.43	90.34	90.07	90.70	90.09	88.51	89.34	90.08	89.07	90.43	90.34	90.07
pl	94.15	94.31	95.28	94.14	93.90	95.34	93.90	93.50	94.95	93.75	93.48	94.94	94.14	93.90	95.34

before inclusion at the different components of the MCCGS model. The remaining training settings are the same, i.e., as mentioned in the parameter settings for the MCCGS model. The reported results in Table 4.3 show that feature inclusion at disambiguator (MCCGS-D) improves the accuracy compared to the rest of the MCCGS for most of the languages. English is the only language that decreases the model performance by 1% after utilizing such features. Thus, Czech is the only language on which feature inclusion at contextual encoder (MCCGS-CE) improves the accuracy by 0.59%. Feature inclusion at input encoder (MCCGS-IE) improves the accuracy of Dutch, Portuguese, Spanish, Danish, French and German languages by 1.87%, 4.64%, 1.78%, 0.92%, 1.52% and 3.14%, respectively. There are four languages (Spanish, Italian, Bulgarian and German) on which the DsDs model performs better compared to the MCCGS model, as interpreted by Table 4.2. However, MCCGS-IE further improves accuracy for Bulgarian and German languages mentioned in Table 4.3.

The proposed model, MCCGS and its variants have been compared with the TC_w, Retro, DsDs and PSA model to show the significance of the results by single factor ANalysis Of VAriance (ANOVA) test. The MCCGS, MCCGS-IE, MCCGS-CE and MCCGS-D have obtained the p-values, 0.003, 0.001, 0.002 and 0.000, respectively. These p-values are comparatively lower than the significance value (0.05), which shows stronger evidence against the earlier hypothesis made by state-of-the-art models. This conclusion supports our obtained results, mentioned in Table 4.2 and Table 4.3.

The mean accuracy of the model compared to their variants and some earlier state-of-the-art models show that MCCGS-D provides the highest score, 92.3%. The proposed MCCGS model provides a mean accuracy of 90.4%, further improved by feature inclusion. The state-of-the-art models provide 83.6%, 81.3%, 84.1% and 90.18% mean accuracy by TC_w, Retro, DsDs and PSA, respectively. The comparison of the mean accuracy among those models is mentioned in Figure 4.5.

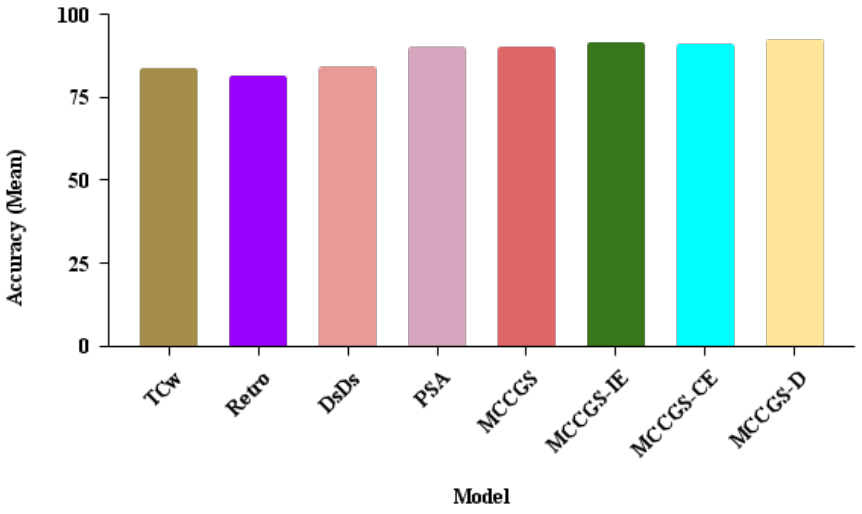

FIGURE 4.5: Comparison of mean accuracy scores

TABLE 4.4: Effect of the chunk size on the model performance in terms of mean accuracy

Chunk Size	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
MCCGS	86.13	86.78	87.51	88.24	88.67	89.12	89.94	90.00	90.19	90.40
MCCGS-IE	86.94	87.83	88.67	89.15	89.85	89.65	89.98	90.51	91.39	91.40
MCCGS-CE	85.03	86.14	87.09	88.40	88.93	89.95	89.47	90.40	90.61	90.90
MCCGS-D	87.67	87.45	88.10	88.95	89.31	90.14	90.87	91.32	91.98	92.30

4.6.3.2 Effect of chunk size

One of the hyper-parameters of the MCCGS model has a chunk size w, mainly used to capture the word information from adjacent words. It allows managing non-continuous relations among the words of a selected chunk. For this claim, we have performed extensive experiments with different chunk sizes for each model. Table 4.4 shows the acquired mean accuracy results of the MCCGS with its variants, MCCGS-IE, MCCGS-CE, MCCGS-D, as the chunk size of 1 to 10. It is empirical proof that chunk size increases will increase the mean accuracy for all the models. This incremental rate has fallen after the window size of 9 due to obtaining consistent accuracy.

4.6.3.3 Experiments on Indo-Aryan languages

To show the effectiveness of our proposed model for low resource languages, the languages which belong to Indo-Aryan languages in the UD treebank has been exploited for the experiments. In this, Marathi (mr) (UFAL treebank), Tamil (ta) (TTB treebank), Telugu (te) (MTG treebank) and Sanskrit (sa) (Vedic treebank) languages with their splits, i.e. 373, 400, 1051, and 2524 sentences of respective languages for the training set and 46, 80, 131 and 1473 sentences of respective languages for the development set have evaluated. Similarly, another treebank based on Paninian Grammar Framework, Hindi-Urdu multi-representational treebank[2] (HUTB), has been evaluated. From this treebank, the Hindi and Urdu model trained on 16997, 5648 sentences and tested on 1910, 635 sentences, respectively [24, 23]. The Hindi treebank dataset belongs to the News and Articles domain.

The MCCGS-D model performs better compared to the rest of the model for all these Indo-Aryan languages, as shown in Table 4.5. However, significant improvements in accuracy have been shown for the Marathi and Tamil languages which are 8.55 and 7.7 for the respective languages, even with minimal training data. This significant improvement also exists for the Precision, Recall, F1 and MCC scores as well. For the Sanskrit language, this improvement is minimal, 1.49 compared to Marathi, Tamil and Telugu. Similarly, the MCCGS-CE model performs adequately on Urdu language of the HUTB dataset compared to other models. Since Hindi from the HUTB dataset has a sufficient number of annotated sentences, the MCCGS model performs better.

4.6.3.4 Analysis

Here we present an analysis of the proposed MCCGS and its variations to understand the influence of feature inclusion better. For this realization, a sample text is taken from the development set of English-EWT treebank for simulating performance, as mentioned in Figure 4.6. Expressing past information of the sentence, the reinstating conjunction 'when' is used to connect two clauses. The subordinate clause 'came' indicates the features as tense being past, mood indefinite, and sentence type is finite. The same features signify

[2]http://ltrc.iiit.ac.in/hutb_release/

TABLE 4.5: Obtained scores on the Indo-Aryan languages of UD treebank and HUTB dataset

		UD treebank (Lang)				HUTB (Lang)	
	Models	**mr**	**ta**	**te**	**sa**	**hi**	**ur**
Precision	MCCGS	80.20	78.56	91.60	96.64	95.68	86.93
	MCCGS-IE	88.16	79.73	94.68	97.91	94.10	84.74
	MCCGS-CE	81.44	79.62	94.12	97.03	94.85	87.49
	MCCGS-D	93.33	87.25	96.02	97.29	94.45	86.95
Recall	MCCGS	85.03	79.80	93.29	95.79	95.60	87.06
	MCCGS-IE	89.09	82.17	95.20	96.23	94.78	85.38
	MCCGS-CE	85.22	80.15	94.20	95.46	94.59	87.62
	MCCGS-D	93.58	87.50	96.69	97.28	94.52	87.48
F1	MCCGS	82.37	77.79	92.38	95.66	95.60	87.06
	MCCGS-IE	87.33	80.18	94.67	96.56	94.78	85.38
	MCCGS-CE	81.50	78.70	93.42	95.54	94.59	87.62
	MCCGS-D	93.09	86.90	96.25	97.27	94.52	87.48
MCC	MCCGS	77.32	76.46	76.58	92.83	94.16	86.67
	MCCGS-IE	73.71	82.02	90.73	94.68	94.23	86.90
	MCCGS-CE	82.34	74.42	92.07	95.75	94.10	88.63
	MCCGS-D	92.99	87.04	96.23	96.73	93.93	86.13
Accuracy	MCCGS	85.03	79.80	93.29	95.79	95.64	86.81
	MCCGS-IE	89.09	79.73	95.20	96.23	94.41	84.24
	MCCGS-CE	85.22	80.15	94.20	95.46	94.57	87.47
	MCCGS-D	93.58	87.50	96.69	97.28	94.25	86.76

the matrix clause 'arrested'. Even though these two are different clauses connected by the conjunction, the model can realize long-distance relations.

According to Greenberg's language universal [85], a noun (NN) is always followed by an adjective (JJ). According to this hypothesis, the training data followed a combination of noun phrases (JJ+NN). As a result, the model predicts JJ+NN in most of the cases instead of JJ+JJ. Figure 4.6 example also shows the same combination, which led to the wrong prediction from the proposed models. It is the same with the verb phrase as well. Verb phrase contains RB+VBN while nouns are directly preceded by the verb in this scenario from the proposed model. The impact of additional features can be seen in the model prediction. The MCCGS model predicts wrong output for adjacent words. For example, 'briefly arrested' should be an adverb (RB) and verb (VDB), but the model predicts adjectives (JJ) and noun (NN), respectively. After using the features during the

model training, we found that the number of errors has drastically improved. Here, the 'arrested' is getting tense information (Tense=Past) from the 'came' and it carries forward in the other morphologically similar words. Due to this, the MCCGS-IE, MCCGS-CE and MCCGS-D predict RB+NN, VBD+VBD and RB+VBD, respectively, for the 'briefly arrested'. The feature inclusion near the label prediction provides an accurate result for capturing longer dependencies among words.

Sentence	In	Fallujah	,	hundreds	of	demonstrators	came	out	against	US	troops	when	they	briefly	arrested	a	young	newlywed	bride	.
Actual Label	IN	NNP	,	NNS	IN	NNS	VBD	RB	IN	NNP	NN	WRB	PRP	RB	VBD	DT	JJ	JJ	NN	.
MCCGS	IN	NNP	,	NNS	IN	RB	VBD	IN	IN	NNP	NN	WRB	PRP	JJ	NN	DT	JJ	JJ	NN	.
MCCGS-IE	IN	NNP	,	NNS	IN	NNS	VBD	IN	IN	NNP	NN	WRB	PRP	RB	NN	DT	JJ	JJ	NN	.
MCCGS-CE	IN	NNP	,	NNS	IN	RB	VBD	IN	IN	NNP	NN	WRB	PRP	VBD	VBD	DT	JJ	JJ	NN	.
MCCGS-D	IN	NNP	,	NNS	IN	NNS	VBD	RB	IN	NNP	NN	WRB	PRP	RB	VBD	DT	JJ	JJ	NN	.

FIGURE 4.6: Example of models prediction to POS tagging

4.7 Experiment-II: Contrastive MCCGS

The majority of standard annotated data for POS tagging belongs to specific domains such as News, being trained on a specific domain, the trained model tested on other domains like Health, Social media, Literature, and Forum understandably gives a worse performance [57]. One major reason is that the source and target vocabularies are different and thus, some target words may never have appeared during the training phase. For instance, in the Hindi treebank dataset[3][4], disease-specific terminology such as disease names, symptom identifiers and treatments are frequent in the Health domain but are rare in the News, Tourism and other domains. In order to generalize models to other domains, an optimal adaptation method is required that can transfer gained knowledge from the high resource domain to the low resource domain as much as possible.

In deep learning, the data is the only source of model training so that the expressivity of the language used depends solely on the domain of the data as far as the modeling process is concerned. That an equivalent performance will be obtained on other domains is not

[3]http://ltrc.iiit.ac.in/hutb_release/
[4]https://ltrc.iiit.ac.in/showfile.php?filename=downloads/kolhi/

guaranteed due to the distinction between the distributions of domain-specific information [27] for different domains. While training the model, the model learns both general and domain-specific features [57]. These general or primitive features can be transferred to the target domain for either extracted features (through Transfer Learning) or initialization (through Fine-tuning). Fine-tuning is very effective as they use the available features from the pre-trained model through neurons [263] and allow their use for other domains to address problems like data scarcity.

When a learned model is employed from a source training data to a different but related target test data, it is pretended that both data accompany an equivalent distribution. When the distribution of both data is distinct, the deep learning model will not generalize well on the target domain. The model learning with the existence of an inequivalent distribution of data is known as Domain Adaptation. Domain adaptation can be accomplished with fine-tuning. The difference of inequivalent distribution is minimized by contrastive learning. Contrastive learning is a self-supervised learning method that learns embedding space by contrasting semantically.

Globerson and Roweis [82] performed adversarial training with missing features where they assumed that missing features are distributed randomly in test data. Søgaard [231] targeted the most predictive features of the model by antagonistic adversaries. This method eliminated the essential features and did a frequent update of rare features (less confident). Guo et al. [90] trained the model by meta-learning in which a single source domain was treated as the meta-target and the remaining as the meta-source. The losses of these meta-learning expertise were minimized based on joint losses. This source and target domain encoding was aligned by adversarial training, where the target domain was treated as unlabelled. There is minimal work reported for POS tagging using adversarial training while used in various other applications associated with domain adaptation [243, 68, 249].

By using the contextual information based on the attention mechanism and contrastive learning to overcome the impact of out-of-vocabulary words while testing the model, MCCGS for low resource language, we add the online noise generated samples into training data and train the model with contrastive learning, which is called contrastive MCCGS (described in section 4.5). In this experiment, data-based supervised domain adaptation

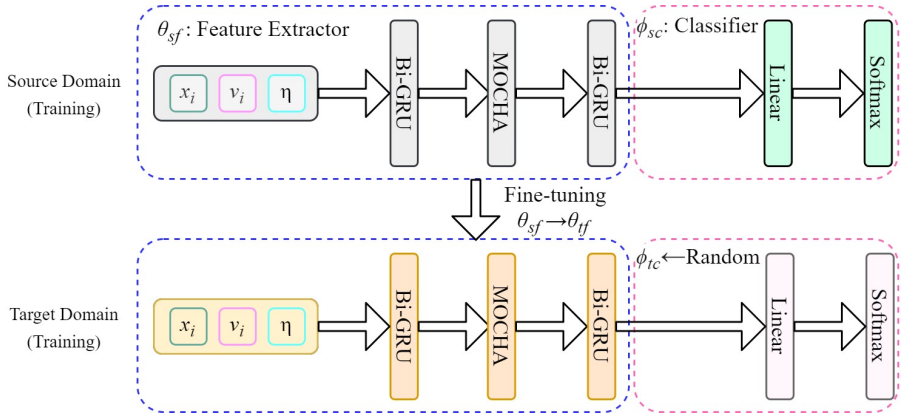

FIGURE 4.7: Domain adaptation

has been followed, where small amounts of annotated datasets that belong to different domains have been used to perform single and multi-source domain adaptation (described in section 4.7.1).

4.7.1 Domain Adaptation

Domain adaptation approaches follow the data or model-oriented techniques with unsupervised, semi-supervised and supervised settings for diverse applications of NLP, including POS tagging. Here, we have exploited the fine-tuning approach for performing domain adaptation, where the learned features from the source domain are transferred at the time of the target domain learning. This feature transfer procedure closely follows the Meftah and Semmar [150] settings. In this procedure, the entire model has been trained on the source domain for POS tagging and learns the optimal learned parameters.

Each domain dataset has different labels, so the training parameters of the proposed model have been interpreted as label-aware parameters (ϕ_c) and data-aware parameters (θ_f), which attain classification and feature extraction, respectively. Feature extractor parameters are considered transferable due to their generality, which considers MOCHA and both Bi-GRU layers. Linear and Softmax layers are considered classification parameters that are task specific. Based on the training parameters, the model has split into two parts: Feature Extractor and Classifier, as shown in the domain adaptation settings in Figure 4.7.

Algorithm 4.2 Domain adaptation for POS tagging

Require: Target domain data, θ_{sf} : Learnt parameters (source domain) from Algorithm 4.1 for Feature Extractor

$\phi_{tc} \leftarrow$ Randomly initialized parameters (target domain) for Classifier
while available target domain data **do**
    $\theta_{tf} \leftarrow \theta_{sf}$
    for each domain data **do**
        $\theta_{s'f} \leftarrow$ Train Algorithm 4.1 with θ_{tf} and ϕ_{tc}
    end for
    $\theta_{sf} \leftarrow \theta_{s'f}$
end while

The learnt parameters of Feature Extractor, θ_{sf} from source domain are used as parameter initialization to Feature Extractor of the target domain, θ_{tf} during domain adaptation. The remaining parameters of Classifier, ϕ_{tc}, is randomly initialized. After initialization ($\theta_{sf} \rightarrow \theta_{tf}$ and $Random \rightarrow \phi_{tc}$), the model starts learning from previously learned features of the source domain, named as single source domain adaptation, which is shown in Figure 4.7. During the training of domain adaptation, the Classifier parameters have trained on the target domain dataset from scratch. The same procedure follows with multi-source settings, where the Feature Extractor parameters have been transferred from first source to second source ($\theta_{sf} \rightarrow \theta_{s'f}$) and second source to target source ($\theta_{s'f} \rightarrow \theta_{tf}$). The pseudo-code of the domain adaptation is described in Algorithm 4.2.

4.7.2 Dataset Description

For most languages, annotation data is not available due to low resource constraints. Moreover, the availability of a domain-specific annotated dataset in these languages is rare. Here, Hindi Treebank (HT) [23], Penn-TreeBank (PTB) [146] of Wall street journal (WSJ), ARK [177] and the recent TweeBank [136] dataset have been exploited. The HT dataset includes Article, Conversation, Disease and Tourism domain with 32, 32, 39 and 46 POS tags. The PTB, ARK and TweeBank include Newswire and Tweet domains with 45, 26 and 18 POS tags. The comparative statistics of training and testing for each dataset with each domain are depicted in Table 4.6.

TABLE 4.6: Dataset statistics for each domain

Dataset	**Domain**	**Training Data**	**Testing Data**	**Total**
HT	Article	15088	1910	16998
HT	Tourism	2400	622	3022
HT	Conversation	1700	496	2196
HT	Disease	900	594	1494
PTB	Newswire	55000	12499	67499
TweeBank	Tweet	2800	750	3550
ARK	Tweet	1700	675	2375

4.7.3 Experimental Settings

Initially, the state-of-the-art (SOTA) models and CMCCGS model have been evaluated on each domain of the datasets. The maximum length of words and sentences has been fixed for training the model, which is 22 and 52, respectively. However, gradient calculation avoided the padded sentences and words, which in turn prevents overfitting. The character vector size 32 is obtained after applying two filters, 64 and 124, each with the size of 3, with a dropout of 30%. The model is trained with the word vector and GRU units of 100 and 128, respectively. The chunk size is 10. As the annotation corpus is tiny, the model tends to overfit quickly. Hence, dropout and early stoppage have been applied with the value of 50% and 5 as patience, respectively. The model has been trained using Adam optimizer with an initial learning rate of 0.008, which further decays over each epoch by 0.005 for Disease and Tourism. It is tuned to 0.01 as the learning rate and 0.007 as the decay rate for Article, Conversation, PTB, ARK and TweeBank. The SOTA models have trained with their default settings.

4.7.4 Results and Analysis

The standard evaluation metrics such as Precision (Pre), Recall (Re), F1-score (F1) and Accuracy (Acc) have been considered to evaluate the proposed model. Table 4.7 and Table 4.8 shows the results obtained from different domains of HT dataset and the PTB,

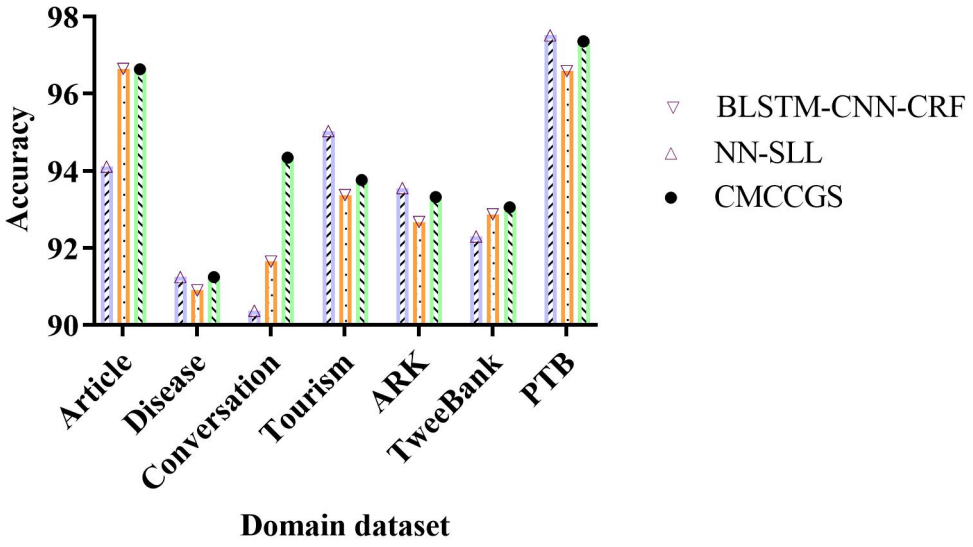

FIGURE 4.8: Accuracy comparison of SOTA and CMCCGS models

ARK and TweeBank dataset, respectively. These results are compared with state-of-the-art models, which are neural network-based sentence-level log-likelihood (NN-SLL) [46] and Bi-directional LSTM-CNNs-CRF (BLSTM-CNN-CRF) [140].

As the size of the Article domain data is large, the CMCCGS model (96.63%) provided a comparable accuracy score to state-of-the-art models, which are 94.11% and 96.64%. The highest score is 91.24% and 94.34% obtained for Disease and Conversation using the CMCCGS model, respectively since the size of the dataset of these domains is minimal. In contrast to the NN-SLL model, the CMCCGS model has reported a lower score, 93.76%, for the Tourism domain. It can be seen that the CMCCGS approach performed better in the Conversation domain. The difference between state-of-the-art (SOTA) models and CMCCGS for the Article domain is very slight, but a significant improvement has been observed for the rest of the domains, as shown in Table 4.7. On the other hand, Table 4.8 showed that PTB and ARK obtained the highest accuracy scores, which are 97.51% and 93.55% through the CMCCGS model, while the BLSTM-CNN-CRF yields 93.05% for TweeBank. It can be clearly interpreted through Figure 4.8. Contrastive training surpasses the performance of current top-performing models [140, 189] for POS tagging in different domains [89, 260]. Hence, the performance is gained on the small size domains after applying the contrastive and adversarial based models.

TABLE 4.7: Results obtained on each domain of HT dataset

Domain	Model	Pre	Re	F1	Acc
Article	NN-SLL [46]	93.93	94.11	93.89	94.11
	BLSTM-CNN-CRF [140]	96.57	96.65	96.57	96.64
	CMCCGS	96.09	96.62	96.35	96.63
Tourism	NN-SLL [46]	95.13	95.03	94.93	95.03
	BLSTM-CNN-CRF [140]	93.42	93.37	93.09	93.37
	CMCCGS	93.76	93.51	93.69	93.76
Conversation	NN-SLL [46]	87.87	90.37	88.84	90.37
	BLSTM-CNN-CRF [140]	91.59	91.65	91.43	91.64
	CMCCGS	94.03	94.35	94.01	94.34
Disease	NN-SLL [46]	91.14	91.24	90.86	91.24
	BLSTM-CNN-CRF [140]	90.74	90.76	90.34	90.89
	CMCCGS	91.91	91.25	91.30	91.24

TABLE 4.8: Results obtained on PTB, TweeBank and ARK datasets

Data	Model	Pre	Re	F1	Acc
PTB	NN-SLL [46]	96.55	96.58	96.53	96.58
	BLSTM-CNN-CRF [140]	97.38	97.36	97.37	97.36
	CMCCGS	97.42	97.51	97.41	97.51
TweeBank	NN-SLL [46]	93.34	92.87	92.92	92.87
	BLSTM-CNN-CRF [140]	93.70	93.70	93.05	93.05
	CMCCGS	93.30	92.30	92.18	92.30
ARK	NN-SLL [46]	93.96	92.67	92.38	92.67
	BLSTM-CNN-CRF [140]	93.48	93.32	93.28	93.32
	CMCCGS	94.32	93.55	93.69	93.55

4.7.4.1 Single-source domain adaptation

It can be understood from Table 4.7 and Table 4.8 that CMCCGS provides robust performance on the minimal datasets. Therefore, the same (hyper-)parameter settings have been employed to perform domain adaptation. The source domain always has larger training data than the target domain. For example, Disease as the target domain considered for Article, Tourism and Conversation domain, Conversation as target domain considered for Article and Tourism, and Tourism as target domain considered for Article domain of HT dataset. The PTB is considered the source domain for ARK and TweeBank. The TweeBank has smaller training data compared to PTB. Thus it is also the target domain for PTB.

The domain adaptation results have been compared with the SOTA model, Hierarchical Bidirectional LSTM-CRF (HBLSTMC)[5] [147]. After domain adaptation, the maximum score has been obtained for Conversation (93.40%) and Tourism (96.76%) by CMCCGS and Disease (95.88%) by HBLSTMC. Similarly, Table 4.10 showed that CMCCGS works adequately for ARK while HBLSTMC model for TweeBank.

The significance of training data to our proposed supervised domain adaptation, CMCCGS model, is observed on the HT dataset in Table 4.9. The Conversation and Disease domain attained the highest score when Article is considered as the source domain. The comparison of Table 4.7 with Table 4.9 gives the significance of domain adaptation where the effect on the performance of Tourism, Conversation and Disease are +3.00%, −0.94% and +2.98%, respectively. On the other hand, the performance has increased by +2.93% for TweeBank and +1.15% for ARK, as can be seen when comparing Table 4.8 with Table 4.10.

TABLE 4.9: Domain adaptation on the HT dataset

Source	**Target**	**Model**	**Pre**	**Re**	**F1**	**Acc**
Article	Conversation	HBLSTMC [147]	90.66	92.35	91.16	92.35
		CMCCGS	92.79	93.41	92.73	93.40
Tourism	Conversation	HBLSTMC [147]	92.51	92.92	92.39	92.92
		CMCCGS	91.79	92.76	91.77	92.75
Article	Disease	HBLSTMC [147]	90.97	91.86	90.50	91.86
		CMCCGS	93.90	94.22	93.87	94.22
Tourism	Disease	HBLSTMC [147]	95.75	95.88	95.75	95.88
		CMCCGS	92.41	92.56	92.28	92.55
Conversation	Disease	HBLSTMC [147]	94.91	94.53	94.52	94.53
		CMCCGS	91.87	91.82	91.60	91.82
Article	Tourism	HBLSTMC [147]	95.16	95.74	95.26	95.74
		CMCCGS	96.64	96.76	96.62	96.76

TABLE 4.10: Domain adaptation on ARK and TweeBank datasets

Source	**Target**	**Model**	**Pre**	**Re**	**F1**	**Acc**
PTB	ARK	HBLSTMC [147]	95.00	94.36	94.67	94.36
		CMCCGS	94.57	94.70	94.54	94.70
PTB	TweeBank	HBLSTMC [147]	96.22	96.23	96.15	96.23
		CMCCGS	95.11	95.23	95.03	95.23

[5]This abbreviation is used to compare our results

4.7.4.2 Multi-source domain adaptation

The deep learning model requires abundant data to provide promising results. The target domain has minimal data in our experiments. To address this, we have used the data from different domains to perform domain adaptation. The Disease domain has limited data compared to the others in the HT dataset and it is considered the target domain and the remaining domains as the source domains for performing multi-source domain adaptation. Similarly, the Tweet domain from ARK is considered the target domain and the Newswire domain from PTB and Tweet domain from TweeBank are considered source domains. Based on the annotation statistics, these datasets can be divided into three categories, namely High (Article, PTB), Moderate (Tourism, Conversation and TweeBank) and Low (Disease and ARK) resources. According to these categories, the following two settings are used:

1. Model training initialized with High category (Article and PTB) and then the learned features were transferred to the Moderate category (Tourism or Conversation and TweeBank), which further applied domain adaptation to the target domain.

2. Model training initialized with High category (Article) and then the learned features were transferred to the Moderate category (Tourism). These learned parameters were then transferred to another Moderate category (Conversation) to apply domain adaptation on the target domain.

These two experiments obtained the best results on the Disease and ARK by using Article with Tourism and PTB with TweeBank as source settings, compared to the others, as shown in Table 4.11.

4.7.4.3 Effect of layers freezing in domain adaptation

The proposed model, CMCCGS, has multiple core layers, i.e., Bi-GRU, MOCHA, and Softmax, which are responsible for capturing different input information. Hence, while performing domain adaptation via the data-based Transfer learning approach, gradual

TABLE 4.11: Results of Multi-source domain adaptation by CMCCGS model

Source (High+Moderate)	**Target**	**Pre**	**Re**	**F1**	**Acc**
Article+Conversation	Disease	92.34	92.82	92.29	92.81
Article+Tourism	Disease	95.21	95.39	95.11	95.38
Tourism+Conversation	Disease	93.35	93.58	93.30	93.58
Article+Tourism+Conversation	Disease	93.39	93.01	93.19	93.01
PTB+TweeBank	ARK	94.29	94.98	94.63	94.98

layers have been fixed to show the model performance. The model layers have been divided into two sub-parts: Layers between the input layer to MOCHA layer, which includes the first Bi-GRU layer, referred to as part(i), and Layers after the MOCHA layer to the inference layer, i.e., the second Bi-GRU, the Dense, and the Softmax layer, referred as part(ii).

The effect on evaluation matrices of both sub-parts being frozen to each domain is shown in Table 4.12. The part(i) layer freezing of domain adaptation for Tourism from Article, for Conversation from Tourism, for ARK from PTB and TweeBank (multi-source) achieved better accuracy than part(ii) layer freezing. The part(ii) layer freezing benefited for Disease from multi-source domain adaptation (Article and Tourism) and TweeBank from PTB. However, the freezing scores of both sub-parts do not yield better results than the full model training while performing domain adaptation, as shown in Figure 4.9.

The training of the two sub-parts being frozen shows up as the differences in the metrics scores for each domain on each kind of domain adaptation as follows:

1. The differences for the Disease domain lie in the ranges of -0.37 to 1.85, -0.23 to 1.74, -0.20 to 1.77 and -0.23 to 1.74 for precision, recall, F1-score and accuracy, respectively.

2. The differences for the Conversation domain are in the ranges of -1.52 to 0.04, -0.71 to 0.02, -1.18 to 0.14 and -0.71 to 0.01 for the respective metrics.

3. Similarly, the Tourism domain has single domain adaptation experiments; hence it yields a difference of $+0.64$ for precision and $+0.12$ for all the remaining metrics.

TABLE 4.12: Results of the domain adaptation based on freezing layers' by CMCCGS model

Source	**Target**	**Freeze**	**Pre**	**Re**	**F1**	**Acc**
Article	Disease	part(i)	92.54	92.96	92.76	92.96
Article	Disease	part(ii)	92.42	92.71	92.24	92.71
Article	Tourism	part(i)	96.40	96.30	96.09	96.30
Article	Tourism	part(ii)	95.76	96.18	95.97	96.18
Article	Conversation	part(i)	90.26	92.15	90.69	92.14
Article	Conversation	part(ii)	91.78	92.25	91.87	92.24
Tourism	Conversation	part(i)	91.66	92.87	91.96	92.86
Tourism	Conversation	part(ii)	91.62	92.85	91.82	92.85
Tourism	Disease	part(i)	94.38	94.51	94.21	94.50
Tourism	Disease	part(ii)	93.48	93.65	93.44	93.64
Conversation	Disease	part(i)	89.64	89.45	89.44	89.45
Conversation	Disease	part(ii)	89.18	88.92	88.81	88.92
Article+Tourism	Disease	part(i)	94.43	94.74	94.45	94.73
Article+Tourism	Disease	part(ii)	94.80	94.97	94.66	94.96
Article+Conversation	Disease	part(i)	91.45	92.07	91.50	92.07
Article+Conversation	Disease	part(ii)	91.09	91.49	91.04	91.48
Tourism+Conversation	Disease	part(i)	92.82	93.07	92.77	93.07
Tourism+Conversation	Disease	part(ii)	90.97	91.33	91.00	91.33
Article+Tourism+Conversation	Disease	part(i)	94.46	94.68	94.36	94.67
Article+Tourism+Conversation	Disease	part(ii)	94.52	94.74	94.42	94.74
PTB	ARK	part(i)	94.65	94.50	94.34	94.50
PTB	ARK	part(ii)	92.37	93.01	92.39	93.01
PTB	TweeBank	part(i)	92.82	92.17	92.07	92.17
PTB	TweeBank	part(ii)	93.98	93.48	93.37	93.48
PTB+TweeBank	ARK	part(i)	94.54	94.51	94.32	94.51
PTB+TweeBank	ARK	part(ii)	93.30	93.59	93.14	93.59

Where negative digits indicate, part(ii) freezing yields a better result than part(i) freezing and vice-versa for positive digits.

4.8 Summary

This chapter demonstrates the effect of contextual information on POS tagging for low resource languages. The attention mechanism – monotonic chunk wise attention – leverages contextual information. That contextual information along with handcrafted feature integration improves the results compared to the existing baselines on UD treebanks and HUTB datasets. The integration of those features at which layer will provide the best

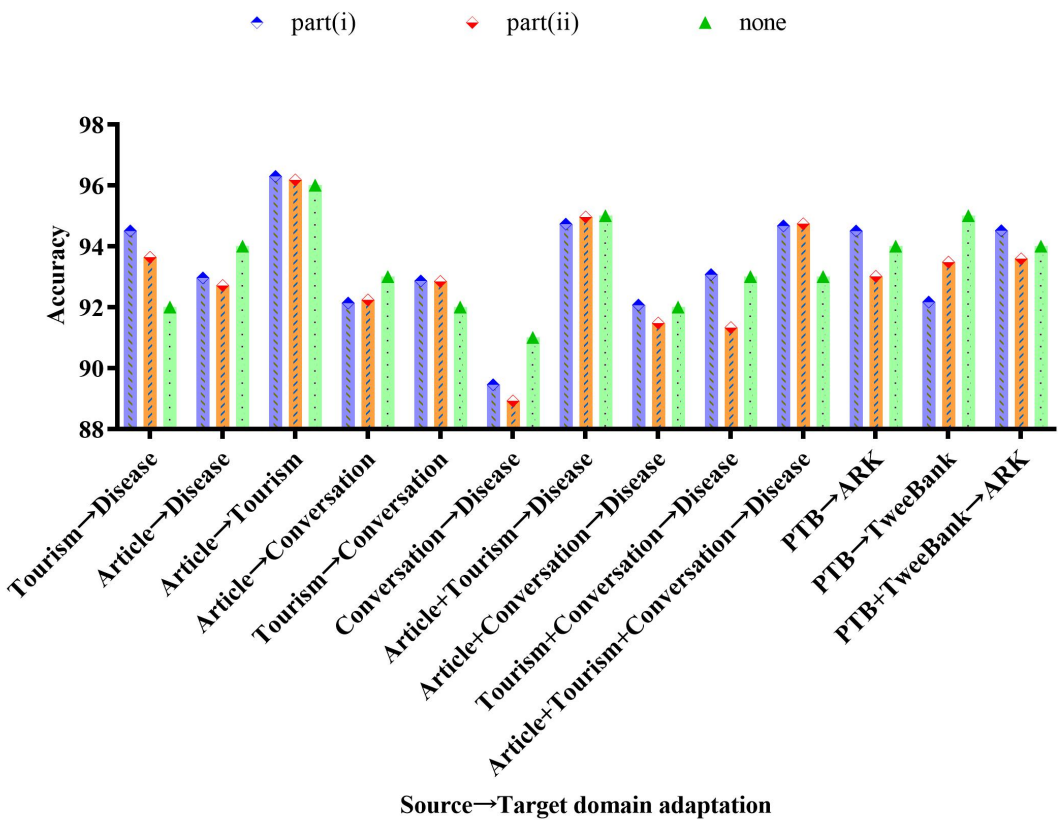

FIGURE 4.9: Accuracy comparison of the CMCCGS model with (part(i) and part(ii)) and without (none) freezing layers'

results is also discussed. It demonstrates that handcrafted features are still useful along with deep learning models for low resource settings. Apart from these, this contextual information with the contrastive learning for domain adaptation aids in improving POS tagging performance. The performed experiments on HT, PTB, WSJ and ARK show that multi-source domain adaptation provides better results compared to single source domain adaptation to target low resource languages, even if source languages could be a combination of low and high resource settings.