Multi-Task Supervised Pretraining for Neural Domain Adaptation

Sara Meftah[◇], **Nasredine Semmar**[◇], **Mohamed Ayoub Tahiri**[◇]
Youssef Tamaazousti[∓], **Hassane Essafi**[◇], **Fatiha Sadat**[+]
[◇]CEA, LIST, Laboratoire Analyse Sémantique Texte et Image, France
[∓]MIT, CSAIL, USA
[+]UQÀM, Montréal, Canada
{firstname.lastname}@cea.fr, ytamaaz@mit.edu, sadat.fatiha@uqam.ca

Abstract

Two main transfer learning approaches are used in recent work in NLP to improve neural networks performance for under-resourced domains in terms of annotated data. 1) *Multi-task learning* consists in training the task of interest with related tasks to exploit their underlying similarities. 2) *Mono-task pretraining*, where target model's parameters are pretrained on large-scale labelled source domain and then fine-tuned on labelled data from the target domain (the domain of interest). In this paper, we propose a new approach that takes advantage of both approaches by learning a hierarchical model trained across multiple tasks from the source domain, then fine-tuned on multiple tasks from the target domain. Our experiments on four NLP tasks applied to social media texts show that our proposed method leads to significant improvements compared to both approaches.

1 Introduction

Deep learning approaches are powerful when dealing with large amounts of annotated data. However, these are only available for a few languages and domains due to the cost of the manual annotation (Duong, 2017). Particularly, despite the valuable importance of Social Media's content for a variety of applications (*e.g.* public security, health monitoring, or trends highlight), this large domain is still poor in terms of annotated data. Furthermore, to build systems for such applications, the machine might understand many low and high-level tasks. Therefore, a model, able to recognise as many linguistic properties as possible from a given sentence, is required.

Many attempts have been done to exhibit the performance of NLP models in low-resource scenarios. Particularly, two dominant approaches of neural Transfer Learning (TL) (Pan and Yang, 2010) are used in the State-Of-The-Art (SOTA): 1) *Mono-Task Pretraining (MTP)*: Sequential Transfer Learning (STL) (Ruder, 2019), performed in two stages: *pretraining* on a rich source-domain on enough training examples and then, *fine-tuning* on the available few target-domain examples. This approach has proved to be powerful in many NLP tasks, outperforming the classic supervised learning paradigm, because it takes benefit from pre-learned knowledge. And 2) *Multi-Task Learning (MTL)* (Caruana, 1997), that showed many benefits in several NLP tasks and applications, consists of training different tasks simultaneously, leveraging learned knowledge from related problems and resulting richer representations with higher generalisation (Collobert and Weston, 2008).

We introduce in this paper a novel method, that we call **Mu**lti-**T**ask **S**upervised **P**re-training and **Ad**aptation (**MuTSPad**), which unifies both approaches discussed above. MuTSPad takes benefit from both, by learning a hierarchical multi-task model trained across multiple tasks from the source-domain, and further fine-tuned on multiple tasks from the target-domain. Hence, in addition to diverse linguistic properties learned from various supervised NLP tasks, MuTSPad takes advantage of the pre-learned knowledge from the high-resource source-domain.

We demonstrate the effectiveness of our approach on domain adaptation from the high-resource News-domain to the low-resourced Tweets-domain. We carry out experiments on four NLP tasks, from the low-level Part-of-Speech tagging (POS) and Chunking (CK) to the higher-level Named Entity Recognition (NER) and Dependency Parsing (DP). MuTSPad exhibits significantly better performance than both TL approaches and is highly competitive compared to best SOTA methods.

Furthermore, to the best of our knowledge, there are no available common datasets containing annotations for all the above-mentioned tasks, neither for the News-domain or the Tweets-domain. Though, many early works had highlighted the intricacy of multi-task training from heterogeneous datasets (Subramanian et al., 2018). Thus, we propose to build multi-task datasets for the News and Tweets domains, by unifying the aforementioned task-independent datasets.

2 Related Work

Our work is related to two lines of research, Sequential Transfer Learning and Multi-Task Learning. In the following, we briefly present the SOTA of each one. Then, we discuss some papers from the literature with a loosely close idea to multi-task pretraining and fine-tuning.

Sequential Transfer Learning (STL) is a TL setting performed in two stages: *Pretraining* and *Adaptation*. The purpose behind using STL techniques for NLP can be divided into two main research areas, "universal representations" and "domain adaptation". The former aims to build neural features transferable and beneficial to a wide range of NLP tasks and domains. *e.g.* ELMo (Peters et al., 2018), BERT (Devlin et al., 2019), etc. The second aims to harness the knowledge represented in features learned on a source domain (high-resourced in most cases) to improve learning a target domain (low-resourced in most cases) (Zennaki et al., 2016, 2019). The source and the target problems may differ on the task, the language or the domain. For instance, cross-lingual adaptation has been explored for sentiment analysis (Chen et al., 2016) and cross-domain adaptation has been applied for POS models adaptation from News to Tweets domain (Gui et al., 2017; Meftah et al., 2017, 2018). Our research falls into the second research area, since we aim, as the last two works, to transfer the knowledge learned when training on the News-domain to improve the Tweets-domain's training.

Multi-Task Learning (MTL) consists in a joint learning of related tasks and thus leverages training signals generated by different tasks (Caruana, 1997). The advantage of using MTL over independent task learning has been shown in many NLP tasks and applications (Lin et al., 2018). The processing (training) order of examples from different tasks (datasets), also called *Scheduling*, is particularly studied in the literature. This could be implicit or explicit (Jean et al., 2018). The formal include, for instance, affecting different learning rates for task-specific parameters. While the second, modify the importance of each task statically or dynamically. *e.g.* Kiperwasser and Ballesteros (2018) proposed variable schedules that increasingly favour the principal task over batches and Jean et al. (2018) proposed adaptive schedules that vary according to the validation performance of each task during training.

Multi-Task Pretraining and Fine-tuning: Multi-task pretraining has been especially explored for learning universal representations (Conneau et al., 2017; Ahmad et al., 2018). Multi-task fine-tuning was recently explored to fine-tune BERT pretrained model in a multi-task fashion on multiple tasks (Liu et al., 2019). Furthermore, in term of using multi-task features for domain adaptation, Søgaard and Goldberg (2016) showed the benefit of multi-task learning for domain adaptation from News-domain to Weblogs-domain for CK task, when disposing CK's supervision only for the source-domain, and lower-level POS supervision for the target-domain. Finally, in terms of unifying multi-task learning and fine-tuning, Kiperwasser and Ballesteros (2018) proposed to improve machine translation with the help of POS and DEP tasks by scheduling tasks during training, starting with multi-tasking of the principal task with auxiliary lower-level tasks (POS and DEP), and as the training graduates, the model trains only to the main task. However, to the best of our knowledge, performing pretraining and fine-tuning on multi-task models for domain adaptation has not been explored in the literature.

3 Model Architecture

3.1 Sequence Labelling Architecture

Regarding the exact architecture of each task, POS, CK and NER tasks are Sequence Labelling (SL) tasks. Given an input sentence of n successive tokens $[w_1, \dots, w_n]$, SL predicts the tag $c_i \in \mathcal{C}$ of every w_i, with $\mathcal{C}$ being the tag-set.

We followed the literature (Ma and Hovy, 2016; Yang et al., 2018) and used a common SL architecture, including three main components: (i) a Word Representation Extractor ($\mathbf{WRE}$), (ii) a Features Extractor ($\mathbf{FE}$) and (iii) a Classifier ($\mathbf{Cl}$). $\mathbf{WRE}$ computes, for each token w_i, a word and a character-level biLSTMs encoder-based embeddings (respectively, $\mathbf{we_i}$=$\mathbf{WE}(w_i)$ and $\mathbf{ce_i}$=$\mathbf{CE}(w_i)$), and concatenates them to get a final representation $\mathbf{x_i}$=($\mathbf{we_i}$,$\mathbf{ce_i}$). $\mathbf{WRE}$'s out-

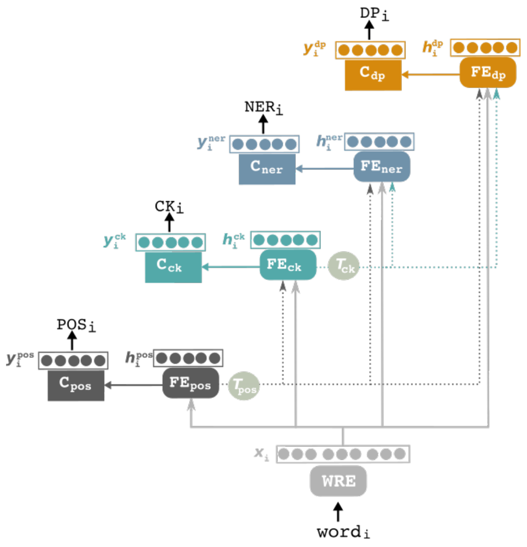

Figure 1: Illustrative scheme of our Hierarchical multi-task architecture.

puts $[x_1, \ldots, x_n]$ are fed into the **FE** that outputs a context sensitive representation for each token, consisting of a single biLSTMs layer which iteratively passes through the sentence in both directions. Finally, **Cl** consists of a fully-connected layer (denoted Ψ) that classifies every given x_i following:

$$\hat{y}_{w_i} = (\mathbf{C} \circ \mathbf{FE} \circ \mathbf{WRE})(w_i). \tag{1}$$

3.2 Dependency Parsing Architecture

For the DP branch, a similar procedure is applied, except that, compared to previous tasks, DP is a a harder problem and thus requires a more complex model. We followed Qi et al. (2018) and used their "neural arc-factored graph-based dependency parser", which is based on the "Deep bi-Affine parser" (Dozat and Manning, 2016)). Indeed, given an input sentence of n successive tokens $[w_1, \ldots, w_n]$, the goal of DP is two folds: 1) identifying, for each w_i, its head $w_j \in S$. The couple of tokens w_i and w_j are called the dependent and the head, respectively. Then, 2) predicting the dependency syntactic relation's class $r_j^i \in \mathcal{R}_{dp}$ relating each dependent-head pair, where $\mathcal{R}_{dp}$ being the dependency-relations set. More precisely, for each token w_i we predict its out-going labelled arc (w_i, w_j, r_j^i). Thus, constructing a syntactic tree structure of the sentence, where words are treated as nodes in a graph, connected by labelled directed arcs.

Hence, as in SL models, the DP architecture is composed of a **WRE** followed by a $\mathbf{FE}^{dp}$ and a $\mathbf{Cl}^{dp}$. Except that $\mathbf{Cl}^{dp}$ consists of *four* classifiers, producing four distinct vectors for representing the word: (i) as a dependent seeking its head; (ii), as a head seeking all its dependants; (iii), as a dependent deciding on its relation; and (iv), as a head deciding on the labels of its dependants. These representations are then passed to biAffine softmax classifiers.

3.3 Hierarchical Multi-Task Architecture

As mentioned above, POS, CK, NER and DP are the four tasks considered in this work. As we aim to learn a multi-task model where the four tasks are learned jointly, the architecture of our model contains a common branch as well as four exits, one per task. Also, as the tasks are hierarchically related to each other, we adopted a *hierarchical* architecture (similar to Hashimoto et al. (2017) and Sanh et al. (2019)). More specifically, we organised the four tasks from low-level to high-level, with each task being fed with a shared word embedding as well as the outputs of all the lower tasks. To construct that hierarchy of tasks, we followed some linguistic hints from the literature. Indeed, many works have shown that POS improves CK (Yang et al., 2017; Ruder12 et al., 2019); NER benefits from POS (Meftah and Semmar, 2018; Ruder, 2019) and CK (Collobert and Weston, 2008); and DP profits from POS and CK (Hashimoto et al., 2017). In simple terms, POS and CK are considered as "universal helpers" (Changpinyo et al., 2018). Thus, based on these linguistic hierarchy observations, we feed POS features to CK; then POS and CK features to both NER and DP.

An illustration of our multi-task hierarchical model is given in Fig.1. More specifically, **WRE** is shared across all tasks, its output (namely $\mathbf{x_i} = \mathbf{WRE}^{shared}(w_i)$) is fed to all branches. The lower component of the POS tagging branch ($\mathbf{FE^{pos}}$) is fed with the shared embedding and after processing, it outputs BiLSTMs features $\mathbf{h_i^{pos}}$. This is then fed into the POS classifier $\mathbf{C^{pos}}$ to calculate predictions through:

$$\hat{y}_{w_i}^{pos} = (\mathbf{C^{pos}} \circ \mathbf{FE^{pos}})(\mathbf{x_i}) \tag{2}$$

These POS features ($\mathbf{h_i^{pos}}$) as well as the shared embeddings $\mathbf{x_i}$ are then fed to the CK branch that outputs a probability distribution for the CK tag-set as follows:

$$\hat{y}_{w_i}^{ck} = (\mathbf{C^{ck}} \circ \mathbf{FE^{ck}})(\mathbf{x_i}, \mathbf{T}^{pos}(\mathbf{h_i^{pos}})) \tag{3}$$

Note that, rather than directly using $\mathbf{FE}^{pos}$'s output, we first reduce its dimensionality by applying a learnable FC layer transformation denoted $\mathbf{T}^{pos}$.

In the same vein, following our hierarchy, the shared embedding, plus the output features of POS ($\mathbf{h_i^{pos}}$) as well as the output features of CK ($\mathbf{h_i^{ck}}$) are fed to the NER branch that outputs one class probability per named entity. Formally, this is computed using:

$$\hat{y}_{w_i}^{ner} = (\mathbf{C^{ner}} \circ \mathbf{FE^{ner}})(\mathbf{x_i}, \mathbf{T}^{pos}(\mathbf{h_i^{pos}}), \mathbf{T}^{ck}(\mathbf{h_i^{ck}})) \quad (4)$$

For DP, similarly to the NER branch, the shared embedding, plus the output features of POS as well as the output features of CK are fed to $\mathbf{FE}^{dp}$, followed by $\mathbf{C}^{dp}$ which outputs:

$$\hat{y}_{w_i}^{dp} = (\mathbf{C^{dp}} \circ \mathbf{FE^{dp}})(\mathbf{x_i}, \mathbf{T}^{pos}(\mathbf{h_i^{pos}}), \mathbf{T}^{ck}(\mathbf{h_i^{ck}})) \quad (5)$$

4 Our Approach

In low-resources scenarios, two approaches are commonly used to alleviate the lack of training data:

1. "Mono-Task Pre-training": pre-training the neural model (supervisedly or unsupervisedly) on a rich source-task before updating its weights on the task of interest (target task);

2. "Multi-task learning": training the task of interest jointly with other auxiliary tasks with labelled data that might force the network to learn useful features.

The first approach is known to work very well since a while, and yielded impressive results in recent years but the last approach seems to struggle as it still faces the problem of annotated data scarcity.

In this paper, to make sure that we learn useful features that are relevant for the tasks of our interest, we propose an approach that combines pretraining and multi-task learning, and thus takes benefits from the rich source-domain, and especially all its available annotated data and tasks. We called our method **Mu**lti-**T**ask **S**upervised **P**re-training and **Ad**aptation (**MuTSPad**). This is described in details in Sec. 4.1, before we describe (in Sec.4.2) how we alleviated the problem of datasets heterogeneity in multi-task learning (*i.e.*, only mono-task labelled datasets are available).

4.1 MuTSPad: Multi-Task Supervised Pretraining and Adaptation

MuTSPad roughly consists in pretraining on a large annotated *multi-task source* dataset and then fine-tuning it on the *multi-task target* dataset, that corresponds to that task of interest. As supervised and unsupervised pretraining, MuTSPad alleviates the lack of annotated data by taking benefit from rich source-domains. However, compared to them it does the pre-training on *multiple* tasks, and not only one. This brings even more real supervision to the network and thus gives more chance to end up with more features. Also important, as source-domains are usually richer than target-domains, we might always find source-datasets that are labelled exactly with all the tasks we want to solve in the target-domain. This enforces the network to learn only features that might be relevant for our tasks of interest, and thus avoid filling up the network with irrelevant features.

More specifically, we considered four commonly used tasks in this work, POS, CK, NER and DP. In classical multi-task scenario all sentences have all needed tasks annotation (i.e. an annotation dataset for each task). However, in our case we have two datasets, the first for News domain and the second for Tweets domain, both are heterogeneous, having one task-annotation per sentence. In contrary to the classical multi-task scenario that assumes having one dataset where words and sentences are labelled for all the tasks, in reality, this is very hard to encounter. Thus, let us consider a more realistic scenario: a set of datasets is given, with each dataset being labelled to only one task. For instance, one dataset is labelled with POS, the other with NER. Note that, though the first is labelled with POS only, it might certainly contain named entities. For this reason, we call this scenario "Heterogeneous multi-task learning". The difficulties of learning in such a scenario are described in details in Sec. 4.2.

Back to MuTSPad, let us assume a set of tasks $\mathcal{T}$, and one dataset $\mathcal{D}_i$ per task $\mathcal{T}_i$. A source multitask model $\mathcal{M}_s$ (described in Fig.1) is first trained on these heterogeneous *source*-datasets $\mathcal{D}^S$. And the set of all parameters learned for each task $\mathcal{T}_i$ as well as task-agnostic ones, are then used to initialise the target multi-task model, denoted $\mathcal{M}_t$. All the weights of this last $\mathcal{M}_t$ are then adapted on the set of target-datasets $\mathcal{D}^T$. Note that, though the sets of target-tasks $\mathcal{T}^T$ and source-tasks $\mathcal{T}^S$ might be the same, their label-sets might differ, thus as in classical fine-tuning, the weights of each task-classifier are randomly initialised. However, for the labels that might be the same, we initialise the weights of the target task-classifiers with those pre-trained on the source-domain.

In terms of loss functions, as in classical multi-task learning, we minimise the weighted sum of each task loss:

$$\mathcal{L} = \frac{\sum_{j=1}^{j=N} \alpha_{task_j} \times \mathcal{L}_{task_j}}{N} \tag{6}$$

Where α_{task_j} represents task-weight and N is tasks number. As we used a hierarchical model for reasons mentioned in Sec. 3.3, we propose to focus the early-stage training on low-level tasks and progressively increase the focus on higher-level ones (along the same line of thought of Kiperwasser and Ballesteros (2018)). However, unlike (Kiperwasser and Ballesteros, 2018) who modified tasks sampling during training, we propose to tune loss calculation minimisation. During training, tasks weights α_{pos}, α_{ck}, α_{ner} and α_{dp} are respectively set up to: $[1, 0.5, 0.25, 0.25]$, and doubled at each epoch until $\alpha = 1$.

4.2 Heterogeneous Multi-Task Learning

As mentioned in the previous section, we mostly face the heterogeneous multi-task learning scenario, where only one task-labels might be assigned to a dataset. In that case, the classical multi-task learning approach is not directly applicable, thus we propose to use the "Scheduling process" (Zaremoodi et al., 2018; Lu et al., 2019) (described in the following paragraph). However, since training with different datasets for each task remains difficult (Subramanian et al., 2018)) we proposed "Dataset Unification" a much simpler and easy to learn method for that scenario.

4.2.1 Tasks Scheduling Procedure

To deal with this heterogeneous aspect, we first use simple frozen uniform scheduling, which we call "one task per batch", similar to (Zaremoodi et al., 2018) and (Lu et al., 2019), where at each iteration of the training process, the task to train is selected randomly. Specifically, the base steps of "one task per mini-batch" scheduling process are as follows: 1) picking a mini-batch of samples from only one particular task and 2) updating *only* the parameters corresponding to the selected task, as well as the subsequent tasks (including the task-agnostic parameters). Thus, at every step only one task is trained. We successively pick all the tasks following a constant ordering strategy "from lower-level to higher-level tasks" (Hashimoto et al., 2017): POS then CK then NER then DP. Thus, every 4 steps, the model sees all the tasks once and learns their corresponding parameters once.

4.2.2 Datasets Unification

To overcome the intricacy of "tasks scheduling process", we propose to construct a *unified dataset* by combining several sources of independent textual annotations. Furthermore, since we are interested in benefiting from pretraining and fine-tuning, we apply unification process on both, source and target-domains.

These datasets contain samples of a broad range of heterogeneous annotations in a variety of contexts (initially sentences are labelled only with one task rather than all), making the multi-task training challenging. Thus, to circumvent this problem, we propose to *unify* the Twitter-domain datasets to form a unified Tweets dataset that we call **TweetAll**. We do the same with standard News-domain datasets to form a unified multi-task dataset that we name **EnglishAll**. Concretely, we enrich the gold annotations of each task with an automatic annotation by applying on its training-set our baseline Mono-Task Learning model of the other 3 tasks. In the end, we obtain two unified datasets one for Tweet (**TweetAll**) and one for English (**EnglishAll**). Thus, in both datasets each sentence is labelled with all tasks (one label is the initial manual annotation and three are generated automatically). Consequently, using our unified datasets brings us to the classical multi-task scenario, where each sentence is annotated with all tasks, thus at each iteration, all tasks are learned and thus all multi-task model's parameters are updated.

5 Experiments

5.1 Domains, Tasks and Datasets

As mentioned above, we conducted experiments on four tasks: two low-level tasks (**POS** and **CK**) and two higher-level ones: (**NER** and **DP**). In terms of domain, data and annotations for the **source-datasets**, we used the *standard English domain* and chose the following datasets: The *WSJ* part of Penn-Tree-Bank (PTB) (Marcus et al., 1993) for POS, annotated with the PTB tag-set; *CONLL2003* for

Task	Classes	Sources	Eval. Metrics	Splits (train - val - test)
POS: POS Tagging	36	WSJ	Top-1 Acc.	912,344 - 131,768 - 129,654
CK: Chunking	22	CONLL-2000	Top-1 Exact-match F1.	211,727 - n/a - 47,377
NER: Named Entity Recognition	4	CONLL-2003	Top-1 Exact-match F1.	203,621 - 51,362 - 46,435
DP: Dependency Parsing	51	UD-English-EWT	Top-1 LAS.	204,585 - 25,148 - 25,096
POS: POS Tagging	17	TweeBank	Top-1 Acc.	24,753 - 11,742 - 19,112
CK: Chunking	18	TChunk	Top-1 Exact-match F1.	10,652 - 2,242 - 2,291
NER: Named Entity Recognition	6	WNUT	Top-1 Exact-match F1.	62,729 - 15,734 - 23,394
DP: Dependency Parsing	51	TweeBank	Top-1 LAS.	24,753 - 11,742 - 19,112

Table 1: Statistics of the datasets we used to train our multi-task learning models. **Top**: datasets of the source domain (called "EnglishAll"). **Bottom**: datasets of the target domain (called "TweetAll").

NER (Sang and De Meulder, 2003); *CONLL2000* (Sang et al., 2000) for CK; and finally *UD-English-EWT* (Nivre et al., 2016) for DP.

In the same vein, for the **target-datasets**, we used the *Tweets domain* and the following datasets: the recent *TweeBank* (Liu et al., 2018) for POS, annotated with the PTB universal tag-set; *WNUT-17* from emerging entity detection shared task (Derczynski et al., 2017) for NER; **TChunk** (Ritter et al., 2011) for CK; and the data annotated with Universal dependency relations in the *TweeBank* dataset for DP[1]. Detailed statistics of all the datasets are summarised in Table 1.

To evaluate our models, we use the accuracy (acc) for POS, Exact-match F1[2] (Li et al., 2020) for NER and CK and labelled attachment score (LAS) (Nivre et al., 2004).

5.2 Comparison methods

5.2.1 Baselines

We compare our method to multiple baselines, that we separate into 4 categories according to the pretraining method.

Without Pretraining: Training from scratch on the Tweets (target-domain) datasets.

- **Mono-Task Learning**: an independent training of our mono-tasks models (one model per task) on every target task separately.
- **Multi-Task Learning**: a *joint training* of our multi-task model described in Sec.3.3 (one model for all the tasks) on all the tasks from target-domain.

Unsupervised pretraining: we replace the **WRE** component in *Mono-Task Learning* by the pre-trained model[3] ELMo (Embeddings from Language Models) (Peters et al., 2018), consisting of a CNNs-based character-level representations followed by a 2-layer LSTMs. Thus, **ELMo** with the randomly initialised **FE** and **Cl** are further trained on the target-domain tasks. Specifically, we run experiments with two ELMo models: 1) $\textbf{ELMo}^{small}$: the small pre-trained model (13.6M parameters) on 1 billion word benchmark. 2) $\textbf{ELMo}^{large}$: the big pre-trained model (93.6M parameters) on 5.5 billion word benchmark.

Supervised pretraining on the source-domain of the network on each task independently then fine-tuning *on the same task* in the Tweets domain. This method is called **Mono-Task Pre-Training**. A variant of it is marked with *, and consists of just pretraining, *i.e.*, without fine-tuning. Note that this variant is possible only when target dataset has the same tagset as the source dataset.

Adversarial pretraining (Ganin et al., 2016; Gui et al., 2017) is particularly used for domain adaptation, that aims to reduce the shift between the source and target domains at the pretraining stage. Precisely, in parallel to task's objective trained on supervised annotations from the source domain, an adversarial objective with respect to a domain discriminator is trained on unsupervised target data to minimise the distance between source and target representations. Followed by a fine-tuning *on the same task* in the Tweets domain.

5.2.2 State-Of-The-Art (SOTA)

We compare our approach to the best SOTA performances for each task:

- **BiAffine** (Dozat and Manning, 2016): we report the LAS score for DP reported by Liu et al. (2018). Note that, in addition to word-level and character embeddings, which we use in our model to represent words, they use predicted POS labels and lemmas as input.
- **Flairs** (Akbik et al., 2019): For NER, using

[1]Note that TweeBank dataset is already anonymised. For TChuck and WNUT datasets, we used simple rules to anonymise usernames and URLs.

[2]SeqEval package were used to calculate F1 metric.

[3]https://allennlp.org/elmo

Method	PreTraining	POS (acc)	DP (LAS)	NER (F1)	CK (F1)	mNRG
SOTA - BiAffine(**Dozat et al., 2017**)	n/a	n/a	77.7	n/a	n/a	n/a
SOTA - PretRand (**Meftah et al., 2019**)	n/a	94.95	n/a	n/a	n/a	n/a
SOTA - Flairs (**Akbik et al., 2019**)	n/a	n/a	n/a	49.59	n/a	n/a
SOTA - MDMT (**Mishra, 2019**)	n/a	92.44	n/a	49.86	87.85	n/a
SOTA - DA (LSTM) (**Gu and Yu, 2020**)	n/a	n/a	n/a	n/a	84.58	n/a
SOTA - DA (BERT$_{BASE}$) (**Gu and Yu, 2020**)	n/a	n/a	n/a	n/a	87.03	n/a
SOTA - DA (BERT$_{LARGE}$) (**Gu and Yu, 2020**)	n/a	n/a	n/a	n/a	87.53	n/a
Best SOTA	n/a	94.95	77.7	49.86	87.85	n/a
Mono-task Learning	none	91.58	67.48	36.75	80.26	0.0
Multi-Task Learning		91.98	71.16	38.98	81.66	5.9
ELMo^{small}	Unsupervised	92.51	69.12	41.57	84.28	9.3
ELMo^{large}		<u>94.02</u>	69.76	**44.95**	<u>85.56</u>	19.9
Mono-Task Pre-Training*	Supervised	n/a	76.92	n/a	70.16	n/a
Mono-Task Pre-Training		93.33	<u>78.21</u>	41.25	84.64	21.5
Adversarial Pre-Training	Adversarial	93.47	77.49	41.68	84.75	22.6
MuTSPad (best)	MultiTask, Sup.	**94.53**	**80.12**	<u>43.34</u>	**85.77**	**31.5**

Table 2: **Results on the TweetAll test-sets for the four tasks**. On the second column, we describe the pretraining type (none, supervised, unsupervised, adversarial and our multi-task supervised). The last column (mNRG that states for median Normalised Relative Gain) aggregates relevantly the scores of the methods across tasks.

a BiLSTM-CRF sequence labelling architecture, fed with Pooled Contextual Embeddings, pre-trained on character-level language models.

- **PretRand** (Meftah et al., 2019): a neural model based on a transfer learning approach, it improves *Mono-Task Pre-Training* baseline by reducing the bias occurring in the pre-trained neurons.

- **Multi-dataset-multi-task (MDMT)** (Mishra, 2019): multi-task learning, based on pre-trained ELMo embeddings, of 4 NLP tasks: POS, CK, super sense tagging and NER, on 20 Tweets datasets 7 POS, 10 NER, 1 CK, and 2 super sense–tagged datasets.

- **Data Annealing (DA)** (Gu and Yu, 2020): a fine-tuning approach similar to *Mono-Task Pre-Training* baseline, but the passage from pretraining to fine-tuning is performed gradually, *i.e.* the training starts with only formal text data (News) at first; then, the proportion of the informal text data (Tweets) is gradually increased during the training process.

5.3 Implementation details

The hyper-parameters (HP) we used are as follows. For **The task-agnostic WRE:** The dimensions of character embedding = 50, hidden states of the character-level biLSTM = 100 and word-level embeddings (updated during training) = 300 (these latter are pre-loaded from Glove pre-trained vectors (Pennington et al., 2014) and fine-tuned during training). For **Sequence labelling branches**: we use a single-layer biLSTM (token-level feature extractor), with dimension = 200. **DP branch HP**: we follow Stanford parser' (//github.com/stanfordnlp/stanfordnlp) HP configuration. **Global HP:** In all experiments, SGD was used for training with early stopping and mini-batches were set to 16 sentences.

5.4 Results

5.4.1 Comparison to SOTA & baselines

Our experimental results are reported in Table 2. Clearly, MuTSPad strongly outperforms baselines, and is very competitive with the best SOTA results. We detail our main findings below:

<u>Multi-Task Learning baseline</u> enhances the performances of all tasks compared to *mono-task learning*. Obviously, it is most benefactor for DP by ∼3.5% since DP is highly influenced by POS labels, while it is least benefactor for POS by ∼0.5%.

<u>Unsupervised pretraining</u>: Clearly, incorporating pre-trained ELMo representations performs better compared to *mono-task learning*. Particularly for NER task with ∼+8% by **ELMo**$^{\mathbf{large}}$. We also found that it improves the other tasks but not with the same order of improvement as for NER, which we mainly attribute to the fact that contextual representations pre-trained on language modelling capture more semantic features. Particularly, we find that DP gains the least from ELMo compared to the other syntactic tasks.

<u>Comparing MuTSPad to baselines</u>: MuTSPad outperforms both TL methods, Multi-Task Learning and Mono-Task Fine-Tuning, on all data-sets, by ∼+26 and ∼+10, respectively, on mNRG (median

Method	POS	DP	NER	CK
w/o unif.	94.08	79.17	**43.34**	84.87
w/ source unif.	94.36	79.67	43.21	**85.77**
w/ source+target unif.	**94.53**	**80.12**	40.65	85.71

Table 3: Impact of Datasets Unification on MuTSPad.

Normalised Relative Gain; a well suited metric for multi-task (Tamaazousti et al., 2019)). Compared to unsupervised pretraining, we can observe that MuTSPad outperforms ELMo on POS, CK and DP, where $\mathbf{Elmo}^{\mathbf{large}}$ brought higher performances for NER. Note that ELMo is complementary to our approach, hence, we expect higher performances when incorporating $\mathbf{Elmo}^{\mathbf{large}}$ to MuTSPad.

Comparing MuTSPad to SOTA: Our score on DP is about ∼2.5% higher than best SOTA scores. For POS, CK and NER experiments, we achieve lower scores than SOTA. It is noteworthy that, first, contrary to our approach, all these methods are mono-task models (except MDMT), *i.e.*, unable to solve other tasks. Second, NER and CK best SOTA used pretrained contextualised representations that harness the performance, namely, Flais embeddings by Akbik et al. (2019), ELMo by Mishra (2019) and BERT by Gu and Yu (2020).

5.4.2 Impact of Datasets Unification

We report in Tab.3 MuTSPad's results:

1) **w/o unif.** : training on independent datasets, using the "one batch per task" scheduling rule. on both stages, pretraining and fine-tuning

2) **w/ source unif.** : In the pretraining stage, training is performed on unified EnglishAll dataset. While in fine-tuning, training is performed on independent datasets.

3) **w/ source+target unif.** : In both pretraining and finetuning stages, training is performed on unified EnglishAll and TwitterAll datasets, respectively.

Clearly, pretraining on unified source datasets (w/source unif) slightly improved performances on all tasks. Nevertheless, finetuning on unified target datasets (w/source+target unif) is beneficial only for POS and DP tasks, while it strongly hurts NER's performance. We mainly attribute this to the NER's "Mono-task learning" model's low performance on Tweets leading to noisy NER automatic predictions. It is noteworthy that, using unified datasets is easier to train, making training convergence faster.

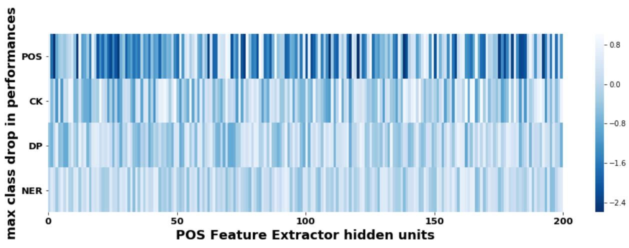

Figure 2: Maximum drop on class-score for each task when ablating individual units from the POS Feature Extractor output ($\mathbf{h}^{\mathbf{pos}}$). Dark/light blue: high/low drop. One can see that it is the POS task that is most impacted by the POS units.

5.5 Low-Level Tasks Importance Analysis

In this section, we investigate how low-level tasks impact high-level tasks in our hierarchical multi-task model (See Fig.1). Specifically, we focus on the impact of $\mathbf{h}^{\mathbf{pos}}$, the representation encoded by the POS task, for CK, NER and DP tasks.

For this purpose, we quantify the importance of $\mathbf{h}^{\mathbf{pos}}$ individual units for POS, CK, NER and DP performances. Assuming that ablating the most important units for a task should bring higher drop in performance compared to the least important units, we perform an individual ablation (also called pruning) of $\mathbf{h}^{\mathbf{pos}}$ units (neurons), as in (Zhou et al., 2018; Dalvi et al., 2019).

Given the already trained target multi-task model $\mathcal{M}_t$, we set the relating weights of each $unit_i$ from $\mathbf{h}^{\mathbf{pos}}$ to zero, *i.e.* $\mathbf{T}^{\mathbf{pos}}$ weights for CK, NER and DP; and $\mathbf{Cl}^{\mathbf{pos}}$ weights for POS. Hence, the ablated unit will not contribute to the final prediction for any input word. Then, with one unit ablated at a time, we launch the inference on each-task's dev-set, then compute the resulting score-drop for each class, leading to a matrix per task $\mathbf{A}^{task} \in \mathbf{M_{d,m}}(\mathbb{R})$, where d is $\mathbf{h}^{\mathbf{pos}}$'s dimension and m is the number of task' classes. This matrix can be summarised in a *max-class-score-drop* vector $\mathbf{v}^{\mathbf{task}} \in \mathbb{R}^d$, where each $\mathbf{v}_{\mathbf{i}}^{\mathbf{task}}$ from the vector represents the max class score drop of $unit_i$ from $\mathbf{h}^{\mathbf{pos}}$.

Applying this method, for POS, CK, NER and DP, leads to 4 *max-class-score-drop* vectors, one for each task, $\mathbf{v_{pos}}$, $\mathbf{v_{ck}}$, $\mathbf{v_{ner}}$ and $\mathbf{v_{dp}}$, that we plot in Fig.4 (one vector per line). We observe high values of *max class score drop* for POS compared to the remaining tasks. First, since $\mathbf{h}^{\mathbf{pos}}$'s units are more important for POS tagging than all other tasks. And, second, $\mathbf{h}^{\mathbf{pos}}$'s units are directly used for prediction for POS while transformed through

Task	Class	Unit	Top-10 activations
CK	B-INTJ	POS-Unit-112	:); awwwwwwww; uggghh; Omg; lol; hahahaha; WELL; Nope; LOL; No
	B-ADJP	POS-Unit-99	rapidely; Deeply; fine; more; hardly; particulary; slower; guilty; loose; entirely
DP	auxiliary	POS-Unit-47	do; can; was; ca; can; 's; would; have; ame; Wo
	discourse	POS-Unit-112	Hhhahahh; no; lmao; sorry; omg; hey; lol; yea; haha; please
NER	B-location	POS-Unit-35	North; Ireland; Italy; Kelly; Qatar; in; southafrica; new; over; Wellington
	B-person	POS-Unit-115	Trilarion; Jo; Watson; Hanzo; Abrikosov; Lily; jellombooty; theguest; Professor

Table 4: Top-10 words activating positively (red) or negatively (blue) (Since LSTMs generate positive and negative activations) some units from $\mathbf{h}^{\mathbf{pos}}$ that are the most important for different classes from CK, DP and NER

several layers for the other tasks. Furthermore, we can also observe that $\mathbf{h}^{\mathbf{pos}}$'s units are more important for CK and DP compared to NER.

Moreover, we attempt to peek inside some units from $\mathbf{h}^{\mathbf{pos}}$, the ablation thereof begets the higher drop in CK, DP and NER classes-scores. Specifically, we report in Tab.4 the top-10 words activating some of these units, as in (Kádár et al., 2017; Meftah et al., 2019). Expectedly, we found that some of POS' units are firing, and thus specialised, on patterns that are beneficial for higher-level tasks. For instance, Unit-99, specialised on adjectives ending with the suffix "ly", is highly important for the CK class "B-ADJP" (*beginning of adjectival phrase*). Also, Unit-115, is firing on persons names, a valuable pattern for "B-person" class of NER. Interestingly, we found some units that are beneficial for multiple tasks, *e.g.* Unit-112, which is specific for interjections, is also important for both "discourse" class for DP and "B-INTJ" (*beginning of an interjection phrase*) for CK.

6 Conclusion

In this research, we have proposed **MuTSPad**, a new approach based on Transfer Learning (TL) for domain adaptation with three main contributions: 1) Consolidating two TL's approaches, sequential transfer learning and multi-task learning, by pretraining on resource-rich domain and fine-tuning on low-resourced domain in a multi-task fashion; 2) Unifying independent datasets to overcome the intricacy of multi-task training from different datasets; and 3) Conducting a set of individual units ablation, refining our understanding on how individual neurons lower-level tasks impact high-level tasks. We showed through empirical results on domain adaptation from *News* to *Tweets* that the proposed method *MuTSPad* allows a simultaneous benefit from similarities between domains and tasks, yielding better transfer learning performances on four NLP tasks, and outperforming the state-of-the-art on Dependency Parsing task.

This study leaves several important open directions for future work. First, we should explore soft multi-task architectures. Second, we expect to explore the combination of supervised and unsupervised multi-tasking. We also plan to explore the benefit of **MuTSPad**'s learned representations for higher-level NLP applications such as machine translation and sentiment analysis.

References

Wasi Uddin Ahmad, Xueying Bai, Zhechao Huang, Chao Jiang, Nanyun Peng, and Kai-Wei Chang. 2018. Multi-task learning for universal sentence representations: What syntactic and semantic information is captured? *arXiv preprint arXiv:1804.07911*.

Alan Akbik, Tanja Bergmann, and Roland Vollgraf. 2019. Pooled contextualized embeddings for named entity recognition.

R Caruana. 1997. Multitask learning [phd dissertation]. school of computer science.

Soravit Changpinyo, Hexiang Hu, and Fei Sha. 2018. Multi-task learning for sequence tagging: An empirical study. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2965–2977.

Xilun Chen, Yu Sun, Ben Athiwaratkun, Claire Cardie, and Kilian Weinberger. 2016. Adversarial deep averaging networks for cross-lingual sentiment classification. *arXiv preprint arXiv:1606.01614*.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364*.

Fahim Dalvi, Nadir Durrani, Hassan Sajjad, Yonatan Belinkov, Anthony Bau, and James Glass. 2019. What is one grain of sand in the desert? analyzing individual neurons in deep nlp models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6309–6317.

Leon Derczynski, Eric Nichols, Marieke van Erp, and Nut Limsopatham. 2017. Results of the wnut2017 shared task on novel and emerging entity recognition. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 140–147.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Timothy Dozat and Christopher D Manning. 2016. Deep biaffine attention for neural dependency parsing. *arXiv preprint arXiv:1611.01734*.

Timothy Dozat, Peng Qi, and Christopher D Manning. 2017. Stanford's graph-based neural dependency parser at the conll 2017 shared task. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 20–30.

Long Duong. 2017. *Natural language processing for resource-poor languages*. Ph.D. thesis.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030.

Jing Gu and Zhou Yu. 2020. Data annealing for informal language understanding tasks. *arXiv preprint arXiv:2004.13833*.

Tao Gui, Qi Zhang, Haoran Huang, Minlong Peng, and Xuanjing Huang. 2017. Part-of-speech tagging for twitter with adversarial neural networks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2411–2420.

Kazuma Hashimoto, Yoshimasa Tsuruoka, Richard Socher, et al. 2017. A joint many-task model: Growing a neural network for multiple nlp tasks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1923–1933.

Sébastien Jean, Orhan Firat, and Melvin Johnson. 2018. Adaptive scheduling for multi-task learning.

Akos Kádár, Grzegorz Chrupała, and Afra Alishahi. 2017. Representation of linguistic form and function in recurrent neural networks. *Computational Linguistics*, 43(4):761–780.

Eliyahu Kiperwasser and Miguel Ballesteros. 2018. Scheduled multi-task learning: From syntax to translation. *Transactions of the Association of Computational Linguistics*, 6:225–240.

Jing Li, Aixin Sun, Jianglei Han, and Chenliang Li. 2020. A survey on deep learning for named entity recognition. *IEEE Transactions on Knowledge and Data Engineering*.

Ying Lin, Shengqi Yang, Veselin Stoyanov, and Heng Ji. 2018. A multi-lingual multi-task architecture for low-resource sequence labeling. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 799–809, Melbourne, Australia. Association for Computational Linguistics.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-task deep neural networks for natural language understanding. *arXiv preprint arXiv:1901.11504*.

Yijia Liu, Yi Zhu, Wanxiang Che, Bing Qin, Nathan Schneider, and Noah A Smith. 2018. Parsing tweets into universal dependencies. In *NAACL*, volume 1, pages 965–975.

Peng Lu, Ting Bai, and Philippe Langlais. 2019. Sc-lstm: Learning task-specific representations in multi-task learning for sequence labeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2396–2406.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *arXiv preprint arXiv:1603.01354*.

Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank.

Sara Meftah and Nasredine Semmar. 2018. A neural network model for part-of-speech tagging of social media texts. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018).*

Sara Meftah, Nasredine Semmar, Fatiha Sadat, and Stephan Raaijmakers. 2018. Using neural transfer learning for morpho-syntactic tagging of south-slavic languages tweets. In *Proceedings of the Fifth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial 2018)*, pages 235–243.

Sara Meftah, Nasredine Semmar, Othman Zennaki, and Fatiha Sadat. 2017. Using transfer learning in part-of-speech tagging of english tweets.

Sara Meftah, Youssef Tamaazousti, Nasredine Semmar, Hassane Essafi, and Fatiha Sadat. 2019. Joint learning of pre-trained and random units for domain adaptation in part-of-speech tagging. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4107–4112.

Shubhanshu Mishra. 2019. Multi-dataset-multi-task neural sequence tagging for information extraction from tweets. In *Proceedings of the 30th ACM Conference on Hypertext and Social Media*, pages 283–284. ACM.

Joakim Nivre, Marie-Catherine De Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, et al. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 1659–1666.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2004. Memory-based dependency parsing. In *Proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL-2004) at HLT-NAACL 2004*, pages 49–56.

Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL*, volume 1, pages 2227–2237.

Peng Qi, Timothy Dozat, Yuhao Zhang, and Christopher D. Manning. 2018. Universal dependency parsing from scratch. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 160–170, Brussels, Belgium. Association for Computational Linguistics.

Alan Ritter, Sam Clark, Oren Etzioni, et al. 2011. Named entity recognition in tweets: an experimental study. In *EMNLP*, pages 1524–1534. Association for Computational Linguistics.

Sebastian Ruder. 2019. *Neural Transfer Learning for Natural Language Processing*. Ph.D. thesis, NATIONAL UNIVERSITY OF IRELAND, GALWAY.

Sebastian Ruder12, Joachim Bingel, Isabelle Augenstein, and Anders Søgaard. 2019. Latent multi-task architecture learning.

Erik F Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. *arXiv preprint cs/0306050*.

Tjong Kim Sang, F Erik, and Sabine Buchholz. 2000. Introduction to the conll-2000 shared task: chunking. In *Proceedings of CoNLL-2000, Lisbon, Portugal*, pages 127–132.

Victor Sanh, Thomas Wolf, and Sebastian Ruder. 2019. A hierarchical multi-task approach for learning embeddings from semantic tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6949–6956.

Anders Søgaard and Yoav Goldberg. 2016. 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 (Volume 2: Short Papers)*, volume 2, pages 231–235.

Sandeep Subramanian, Adam Trischler, Yoshua Bengio, and Christopher J Pal. 2018. Learning general purpose distributed sentence representations via large scale multi-task learning. *ICLR*.

Youssef Tamaazousti, Hervé Le Borgne, Céline Hudelot, Mohamed El Amine Seddik, and Mohamed Tamaazousti. 2019. Learning more universal representations for transfer-learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Jie Yang, Shuailong Liang, and Yue Zhang. 2018. Design challenges and misconceptions in neural sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING)*.

Zhilin Yang, Ruslan Salakhutdinov, and William W Cohen. 2017. Transfer learning for sequence tagging with hierarchical recurrent networks. *arXiv preprint arXiv:1703.06345*.

Poorya Zaremoodi, Wray Buntine, and Gholamreza Haffari. 2018. Adaptive knowledge sharing in multi-task learning: Improving low-resource neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 656–661.

Othman Zennaki, Nasredine Semmar, and Laurent Besacier. 2016. Inducing multilingual text analysis tools using bidirectional recurrent neural networks. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 450–460.

Othman Zennaki, Nasredine Semmar, and Laurent Besacier. 2019. A neural approach for inducing multilingual resources and natural language processing tools for low-resource languages. *Natural Language Engineering*, 25(1):43–67.

Bolei Zhou, Yiyou Sun, David Bau, and Antonio Torralba. 2018. Revisiting the importance of individual units in cnns via ablation. *arXiv preprint arXiv:1806.02891*.