Proceedings of the LREC 2020 – 4th Workshop on Computational Approaches to Code Switching, pages 17–25
Language Resources and Evaluation Conference (LREC 2020), Marseille, 11–16 May 2020
European Language Resources Association (ELRA), licensed under CC-BY-NC

When is Multi-task Learning Beneficial for Low-Resource Noisy Code-switched User-generated Algerian Texts?

Wafia Adouane and Jean-Philippe Bernardy
Department of Philosophy, Linguistics and Theory of Science (FLoV),
Centre for Linguistic Theory and Studies in Probability (CLASP), University of Gothenburg
{wafia.adouane, jean-philippe.bernardy}@gu.se

Abstract

We investigate when is it beneficial to simultaneously learn representations for several tasks, in low-resource settings. For this, we work with noisy user-generated texts in Algerian, a low-resource non-standardised Arabic variety. That is, to mitigate the problem of the data scarcity, we experiment with jointly learning progressively 4 tasks, namely code-switch detection, named entity recognition, spell normalisation and correction, and identifying users' sentiments. The selection of these tasks is motivated by the lack of labelled data for automatic morpho-syntactic or semantic sequence-tagging tasks for Algerian, in contrast to the case of much multi-task learning for NLP. Our empirical results show that multi-task learning is beneficial for some tasks in particular settings, and that the effect of each task on another, the order of the tasks, and the size of the training data of the task with more data do matter. Moreover, the data augmentation that we performed with no external resources has been shown to be beneficial for certain tasks.

Keywords: Algerian Arabic, code-switched user-generated data, multi-task learning, low-resource colloquial languages

1. Introduction

New breakthrough results are continuously achieved for various natural language processing (NLP) tasks, often thanks to the availability of more data and computational power. Likewise, various learning frameworks have been proposed for NLP including multi-task learning. Multi-task learning is about transferring knowledge learned in one task to other tasks by sharing representations (Caruana, 1997). The assumption is that the final learned shared representations are conditioned on the multiple tasks learned simultaneously, and as such they generalise better compared to separate training for each task. This works well when the jointly learned tasks are beneficial for each other, or in cases where a well-performing (auxiliary) task with large data is trained with a related (target) task with less data. However, predicting when tasks are useful for each other remains an open theoretical question and the reported results are still experimental.

This paper is an attempt to take advantage of the state-of-the-art advances in NLP, namely deep neural networks (DNNs) and multi-task learning in order to mitigate the problem of the scarcity of labelled data for colloquial Algerian language (henceforth referred to as ALG). Our main contributions are (1) the creation of a new dataset for code-switched Named Entity Recognition for ALG. (2) An investigation of the settings where it is beneficial to share representations learned between two or several tasks. To this end, we jointly train 4 tasks (or subsets thereof): (1) Code-Switch Detection (CSD), (2) Named Entity Recognition (NER) —both framed as sequence tagging— (3) Spelling Normalisation and Correction (SPELL) —framed as a sequence-to-sequence task— and (4) identifying users' sentiments (SA) —framed as a classification task.

We analyse (1) the effect of each task on another, (2) whether task order matters or not, (3) whether word context for the sequence-to-sequence task is important or not, (4) whether the size of the training data of the task with more data matters, and (5) whether it is useful to augment the training dataset of sequence-to-sequence task (while not requiring any extra resources). We believe that this investigation will extend the utility of multi-task learning in low-resource settings, particularly for code-switched user-generated data. In our experiments we increase the difficulty of the tasks gradually, for instance learning the tasks in pairs, 3 tasks, then 4 tasks, and increase the size of the training data for SPELL progressively.

The paper is organised as follows. In Section 2 we review related work. In Section 3 we describe our tasks and their corresponding datasets. In Section 4 we present the architecture of our model. In Section 5 we describe our experiments and discuss the results. In Section 6 we conclude with the main findings and outline potential directions for future improvements.

2. Related Work

In general, in the context of multi-task learning, the definition of a task is vague: it can refer to an NLP task (Martínez Alonso and Plank, 2017), to a domain (Peng and Dredze, 2017) or to a dataset (Bollmann et al., 2018). Multi-task learning has been applied successfully to a variety of NLP tasks[1] (Collobert and Weston, 2008; Luong et al., 2016; Martínez Alonso and Plank, 2017; Bingel and Søgaard, 2017), focusing on examining the effect of different auxiliary tasks on the performance of a target task. Changpinyo et al. (2018) use joint learning of 11 sequence tagging tasks, investigating whether doing so benefits all of the tasks. Based on the previously reported results, multi-task learning is a promising framework to improve learning with scarce data. Nevertheless, previous work has been mostly limited to morpho-syntactic and semantic sequence labeling tasks, *inter alia*, part-of-speech tagging, syntactic chunking, supersense tagging, semantic trait tagging, semantic role labeling, semantically related words, as well as multi-perspective question answering, and named entity recognition.

[1]We cite here only a few examples.

But what about the languages (domains) for which we do not have labelled data for morpho-syntactic and semantic tasks? Unfortunately many languages (or domains like user-generated data) do not have labelled data to perform such tasks. Indeed, NLP research is still focused largely only on a few well-resourced languages, and models are trained primarily on large well-edited standardised monolingual corpora, mainly due to historical reasons or current incentives. Additionally, in many cases the developed techniques fail to generalise (Hovy and Spruit, 2016), even to new domains within a single language (Jørgensen et al., 2015), mostly because they are designed to deal with particularly structured corpora.
Accordingly, it is not clear whether the previously reported results using multi-task learning for NLP generalise to low-resource settings. In this work, we begin to answer this question by applying multi-task learning to user-generated data. As a case study, we take the language used in Algeria (ALG) which uses code-switching, non-standardised orthography as well as it suffers from the lack of any NLP processing tools such as a tokenizer or morpho-syntactic parsers. Like Changpinyo et al. (2018), we examine the settings in which our tasks benefit from multi-task learning, including pairwise tasks, order of the tasks and the size of the training data for the task with more data.

3. Tasks and Datasets

3.1. Tasks

In multilingual societies people have access to many linguistic codes at the same time. In diglossic situations people have access to even different linguistic levels of the same language (Major, 2002). It is the case in North Africa, where for historical reasons many languages and language varieties are used simultaneously to various extents, including mostly Berber, Arabic and French (Sayahi, 2014). These languages and language varieties coexist throughout the region and they are actively used on a daily basis (Rickford, 1990). Consequently in speech-like communications, such as in social media, people tend to mix languages.

- **CSD** the task deals with the detection of the language (in multilingual CSD) or language variety (in diglossic CSD) of each word in its context for disambiguation (Elfardy et al., 2013; Samih and Maier, 2016; Adouane and Dobnik, 2017). This is challenging for ALG, because the same script is used for all languages (MSA, local Arabic varieties, Berber, French, and English). To further complicate matters, vowels are omitted from the text.

  On the other hand, the enormous spelling variations in user-generated data for all languages and language varieties (Eisenstein, 2013; Doyle, 2014; Jørgensen et al., 2015) challenges the standard language ideology with regards to whether human languages are universally standardised and uniform (Milroy, 2001). It also poses serious challenges to the current NLP approaches at all linguistic levels.

- **SPELL** the task aims at reducing orthographic variation and noise in the data, by context-dependent spelling correction and normalisation. Indeed, user-generated content in colloquial languages contains lots of spelling variations because these languages do not have standardised orthography and the content is unedited. We stress that SPELL is different from a usual spelling error correction task in that it deals with a non-standardised code-switched language —there is no existing largely agreed on reference spelling (Adouane et al., 2019).

- **SA** the task deals with identifying users' sentiments from their generated comments.

- **NER** the task deals with the detection and classification of mentions referring to entities into pre-defined classes (person, location, organisation, product, company, etc.).

3.2. Datasets

For each task we use a separate labelled dataset. Table 1 shows statistics about the CSD, SA and NER datasets. We give more details for each dataset below.

CSD		SA		NER	
Class	Total	Class	Total	Class	Total
ALG	118,942	MIX	11,736	OOO	67,7191
MSA	82,114	POS	10,698	PER	7,262
FRC	6,045	NEU	7,262	LOC	4,641
BOR	4,025	NEG	6,424	PRO	3,682
NER	2,283			OTH	901
DIG	1,394			ORG	399
SND	687			COM	248
ENG	254				
BER	99				

Table 1: Statistics about the datasets: CSD (#tokens), SA (#samples) and NER (#mentions).

- **CSD** we use the dataset described by Adouane and Dobnik (2017) which consists of 10,590 user-generated texts labelled at a token level (intrasentential), and includes 9 classes: Local Algerian Arabic (ALG), Berber (BER), French (FRC), English (ENG), Modern Standard Arabic (MSA), and Borrowing (BOR) (which refers to foreign words adapted to the Algerian Arabic morphology), Named Entity as a general class (NER), interjections/sounds (SND) and digits (DIG).

- **SPELL** we use the dataset described in (Adouane et al., 2019) which consists of a parallel corpus with 50,456 words and 26,199 types to be corrected or normalised.

- **SA** we use the dataset described by Adouane et al. (2020) which consists of 36,120 user-generated comments labelled for 4 sentiment classes: positive (POS), negative (NEG), neutral (NEU) and mixed (MIX).

- **NER** we could not get any dataset labelled for NER for ALG that would serve directly our purpose. Therefore we compiled a new dataset by combining the two datasets used for CSD and SA, resulting in 46,710 user-generated comments in total. Then with the help of two other native speakers, we manually labelled it for NER task by classifying every named entity mention in one of the 6 pre-defined classes, following the labelling schema used in OntoNotes Release 5.0 [2]. The classes are: person (PER), location (LOC), product (PRO), organisation (ORG) and company (COM). We tagged the rest of named entity mentions like time and events as "other" (OTH) to distinguish them from non-named entities (OOO). In order to identify multi-word expressions as one named entity chunk, we use the IOB (Inside-Outside-Beginning) labelling scheme. The newly labelled corpus for NER task has 17,133 named entities with IOB details.

4. Models

4.1. CSD and NER

We frame CSD and NER as sequence tagging tasks, i.e., the task is to assign one of the pre-defined tags to each token in an input sequence. We use an encoder-decoder architecture similar the one described by Adouane et al. (2018). However, here the encoders are shared between the tasks, while decoders are task-specific.

- The **Token-level encoder** (in dark orange in Figure 1) encodes the input sequence at the token level. It maps each of 430 possible characters (including special characters and emoticons) to a 100-dimensional representation. It is composed of two convolution layers with 100 features and a filter size of 5 with a dropout rate of 20%, followed by ReLU activation and max pooling in the temporal dimension. In sum, it reads an input sequence character by character and outputs character embeddings for each token (constructs token representations).
- The **Sequence-level encoder** (in light orange) acts at a sequence level. It takes the outputs of the token-level encoder (character embeddings) and outputs word embeddings as a representation for the entire sequence. It consists of two convolution layers with 200 features for the first and 100 for the second, a filter size of 3, ReLU activation and a dropout rate of 5%.
- The **Dense layer** (in dark green for CSD and light blue for NER) with softmax activation maps the output of the sequence-level encoder (word embeddings) to CSD or NER tag sets respectively.

4.2. SPELL

We frame SPELL as a sequence-to-sequence prediction task where the input is a user-generated sequence (text) and the output is its normalised and corrected version (sequence). For this, we use an encoder-decoder architecture (Cho et al., 2014) similar to the one described by Adouane et al. (2019).

- The **Encoder** consists of the shared layers described above in 4.1.
- The **Decoder** (in light green) and consists of one Long Short-Term Memory (LSTM) layer (Hochreiter and Schmidhuber, 1997). It takes the output of the sequence-level encoder (word embeddings) as input and reads it character by character. It has a vocabulary size of 430, 100 units, a token representation size of 100 and a dropout rate of 10%. It is followed by a dense layer (in light green too).

4.3. SA

We frame SA as a text classification task: i.e., assign one of the pre-defined tag sets to an input sequence of any length. We use the model described in (Adouane et al., 2020) which consists of two sub-neural networks.

- The **Encoder** consists of the shared layers described earlier in 4.1, namely the Token-level and the Sequence-level encoders.
- The **Dense layer** (in yellow) with softmax activation maps the output of the sequence-level encoder to SA tags.

All models are trained end-to-end for 50 epochs using a batch size of 64 and Adam optimiser. Gradients with a norm greater than 5 are clipped. As the main focus of the multi-task learning, models share embedding and encoder parameters. Each task is run for a full epoch before switching to the next task. Therefore there is no special code to combine losses (each loss function remains the same for a whole epoch).

5. Experiments and Results

In order to evaluate the performance of our model, we shuffled the datasets and split them (with no overlapping parts) as follows.

For **CSD** we use 30% (3,177 samples) as a test set, 10% (1,059 samples) as a development set, and the remaining 60% (6,354 samples) as a training set.

For **SPELL** we use 20% (37,041 samples) as a test set, 5% (9,261 samples) as a development set, and 75% (138,917 samples) as a training set.

For **SA** we use 17% (6,122 samples) as a test set, 10% (3,612 samples) as a development set, and 73% (26,386 samples) as a training set.

For **NER** we use 30% (14,013 samples) as a test set, 10% (4,671 samples) as a development set, and the remaining 60% (28,026 samples) as a training set.

Note that all datasets are separate and are labelled for different tasks using different tag sets (depending on the task). The hyper-parameters mentioned in Section 4 are fine-tuned on the development sets. Given the small size of the CSD dataset and the high sparsity of the SPELL dataset, after fixing the hyper-parameters, we train both on the training and the development sets, following Yin et al. (2015).

To examine the effect of jointly learning the tasks, we experiment with the following setups:

1. **Pairwise tasks**: To measure the effect of a task on a single other task, we train them two at a time, as shown in Table 2.

[2]`https://catalog.ldc.upenn.edu/docs/LDC2013T19/OntoNotes-Release-5.0.pdf`

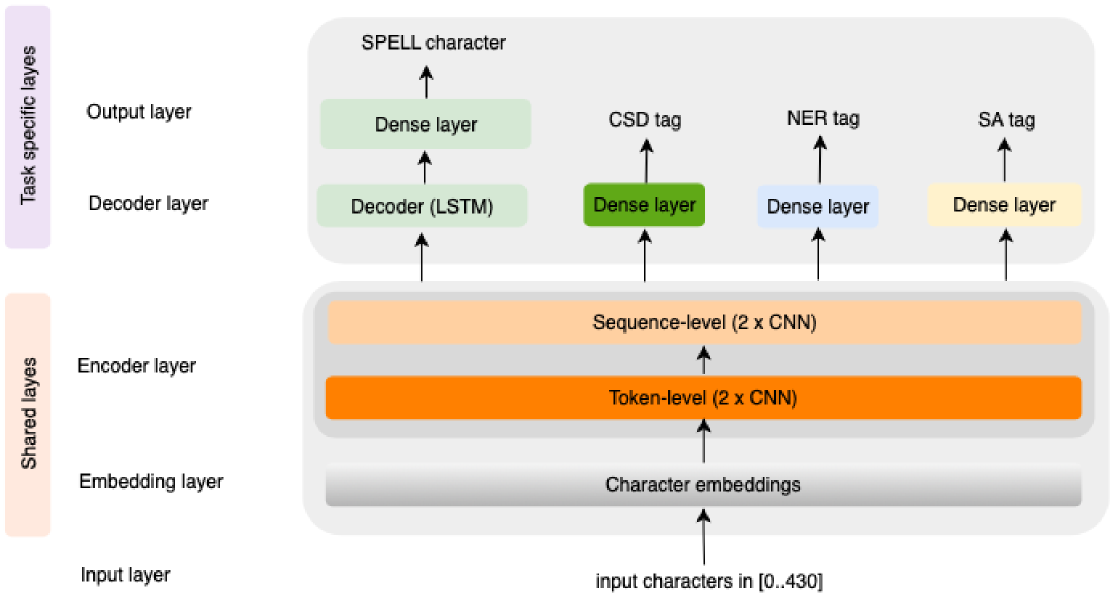

Figure 1: Multi-task model architecture.

2. **Order of tasks**: To check whether the order of tasks affects the overall performance, we run sets of 3 and 4 tasks in various orders. We report the cases where the order has a measurable effect (positive or negative) on the performance.

3. **Context of words**: We are interested in measuring the effect of the context for SPELL (sequence-to-sequence). To do so we either feed the data word by word or whole user-generated text at a time. In the following SPELL will refer to the context-aware task, and SPELL-token refers to the contextless task.

4. **Size of SPELL training data**: We want to investigate the impact of the size of the training data, especially considering that one of the tasks (SPELL) has much more data than the other (CSD, SA and NER) tasks. To do so, we vary only the size of the training data of SPELL while keeping the training sets of CSD, SA and NER fixed each time (as well as the test sets).

5. **Training data augmentation**: We experiment with augmenting the training data for the SPELL task (further referred to as augmented). In this experiment the training data is a combination of tokens and sequence of tokens. (This is equivalent to jointly training SPELL and SPELL-token.)

For each case we take models trained separately (single tasks) as baselines. For **pairwise tasks** we report the detailed results measured as the average Accuracy and macro F-score on the test sets over 50 epochs, thus taking into account the speed of learning. For other experiments (2, 3, 4, and 5) we show the performance, measured as the overall Accuracy, of jointly learning the tasks at hand on the test sets over 20 epochs (we found no significant gain when training for longer and do not report further).

5.1. Pairwise tasks

Task	Tasks	Training	Accuracy (%)	Macro F-score
CSD	CSD	single	96.80	64.54
	CSD + SPELL	joint	96.32	62.27
	CSD + SA	joint	94.30	34.61
	CSD + NER	joint	**97.20**	**71.29**
SPELL	SPELL	single	93.49	
	SPELL + CSD	joint	**93.60**	
	SPELL + SA	joint	93.20	
	SPELL + NER	joint	**93.71**	
SA	SA	single	61.23	54.08
	SA + CSD	joint	**61.35**	53.31
	SA + SPELL	joint	60.74	51.50
	SA + NER	joint	59.82	53.46
NER	NER	single	99.80	49.68
	NER + CSD	joint	**99.82**	48.65
	NER + SPELL	joint	99.78	42.05
	NER + SA	joint	99.74	34.60

Table 2: Macro-average performance of the tasks trained separately and pairwise. Underlined values are baselines. Values in bold show positive effect of jointly learning the tasks at hand.

In Table 2, results measured as Accuracy indicate that learning SPELL, SA and NER tasks jointly with CSD improves their performance over learning them separately — by comparing the performance of single tasks to their performance when jointly trained with CSD.
Note that the gain is mutual between CSD and NER, i.e., jointly learning the tasks benefits both, to different extents. Nevertheless, SPELL and SA slightly benefit from CSD but do not improve it. Interestingly whenever multi-task includes SPELL or SA tasks, the overall performance of the second task (CSD or NER) drops compared to learning the

task separately.

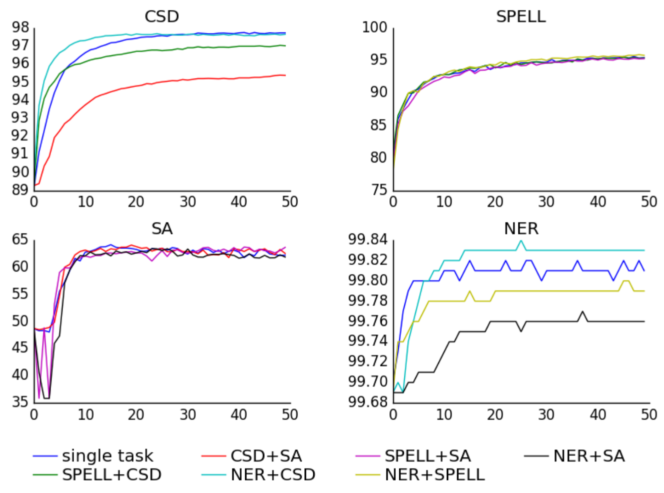

Figure 2: Accuracy (%) of jointly learning 2 tasks for 50 epochs.

A closer look at the results per epoch in Figure 2 indicates that when beneficial, multi-task learning speeds up the performance of the tasks for the first few epochs.

The same behaviour is observed in experiments below (Section 5.2 for instance). This could be because (1) the generated shared representation is not wide enough to capture all tasks perfectly —it needs more parameters in shared layers, or that (2) each task has enough data in itself to reach maximum accuracy. (3) Another hypothesis, which contradicts (2), is that the sparsity and noise in the SPELL and SA training data effects negatively the other tasks.

Jointly training NER with CSD (in turquoise) outperforms training the tasks separately. Furthermore, jointly learning SA with CSD (in red) and SA with SPELL (in pink) outperforms SA trained as a single task. These observations refute hypothesis (2). We controlled for hypothesis (1) by increasing the number of features in the shared layers. That is to say, we tried different values and found that using 500 features in the CNN layers of the token-level encoder, and 500 and 1,000 features for the first and the second CNN layers of the sequence-level encoder has slightly improved the performance of SPELL. However, the overall behaviour of jointly learning SPELL or SA with CSD or NER is still the same. This means that hypothesis (1) does not hold, i.e., it is likely that the noise and sparsity of the SPELL and SA datasets have negative effects on training them jointly with each other or with CSD and NER. Evaluating this hypothesis requires further investigation, which we leave it as future work.

We provide in Table 2 the macro-average F-score for each setting which also reflects the overall impact of jointly learning the tasks by treating all the classes equally. Moreover, since all our datasets are imbalanced both in terms of class distributions and dataset sizes (certain classes have more samples than others and some datasets are much larger than others) we also show the micro F-score at a convergence point for each setting to better analyse the effect of jointly learning the tasks on each class.

Task	Class	CSD	CSD + SPELL	CSD + SA	CSD + NER
CSD	ALG	92.05	89.82↓	83.90↓	91.86↓
	BER	74.29	71.43↓	00.00↓	64.71↓
	BOR	77.10	62.45↓	20.91↓	72.22↓
	DIG	99.93	99.93	99.25↓	99.93
	ENG	26.67	15.38↓	00.00↓	37.50↑
	FRC	83.62	74.45↓	44.93↓	82.17↓
	MSA	90.76	87.88↓	81.71↓	90.39↓
	NER	58.62	26.74↓	02.35↓	62.83↑
	SND	96.14	95.98↓	80.37↓	95.58↓
	Class	SA	SA + CSD	SA + SPELL	SA + NER
SA	MIX	60.38	62.48↑	64.20↑	60.70↑
	NEG	41.72	42.44↑	31.88↓	48.21↑
	NEU	53.80	50.95↓	54.95↑	56.11↑
	POS	75.59	75.92↑	76.92↑	75.48↓
	Class	NER	NER + CSD	NER + SPELL	NER + SA
NER	COM	21.54	29.55↑	11.32↓	00.00↓
	LOC	80.77	81.50↑	74.03↓	66.05↓
	OOO	99.50	99.59↑	99.45↓	99.42↓
	ORG	09.57	06.67↓	03.87↓	00.00↓
	OTH	26.39	27.41↑	22.66↓	18.75↓
	PER	63.38	69.77↑	54.36↓	52.17↓
	PRO	57.20	59.93↑	54.51↓	47.80↓

Table 3: Micro F-score of the tasks in single and multi-task settings. ↑ marks positive effect and ↓ marks negative effect of jointly learning the 2 tasks at hand.

Results in Table 3 show that jointly training CSD with SPELL or SA has negative effect on all CSD classes (marked with ↓). The negative effect of SA is more pronounced. Minority classes (BER, BOR, ENG, FRC, and NER) are more affected than others. Training CSD with NER has also caused some loss in the performance of some classes of CSD (marked with ↓), but the loss is smaller than when trained with SPELL or SA. The positive effect of NER task on CSD (marked with ↑) could be attributed to its improvements for ENG and NER classes (two minor classes) with a gain of 10.83 and 4.21 points on the F-score respectively. One possible explanation for this improvement could be that the model could extract some underlying structures between some named entity mentions and English words used in the same context. It could be also that it becomes easier for the model to further classify a token in one of NER classes when it knows it is a mention of a named entity.

As shown in Table 3, some classes are harder to learn than others, single trained models struggle also with them. Overall SA benefits from CSD and NER. On the one hand, the gain from CSD could be attributed to its positive effect on MIX, NEG and POS classes. Nevertheless, CSD has a negative effect on NEU with a loss of 2.85 points on the F-score. On the other hand, NER has improved MIX, NEG and NEU classes with a slight loss on POS. SPELL has improved MIX, NEU and POS and caused significant drop on NEG with a loss of 9.84 points on the F-score.

The main difference between the effect of the tasks is mainly on the minority classes (NEG and NEU). This suggests that the tasks could be complementary and their effect could be optimised if trained jointly. This is confirmed when training the 4 tasks together as shown in Figure 4 —at least for the first 10 epochs for SA.

SPELL and especially SA have a significant negative impact on all classes of NER. Nonetheless CSD has improved all NER classes except ORG (which a single NER model struggles to capture, with an F-score of only 9.57).

5.2. Order of tasks

Results in Figure 3 show that, except for NER, jointly learning the CSD, SPELL and SA tasks improves their performance over learning each one separately (as single tasks) only for the first few (7) epochs: after that, learning CSD as a single task outperforms training it with other tasks (blue line), and the effect of learning jointly the tasks is not clear for SPELL and SA.

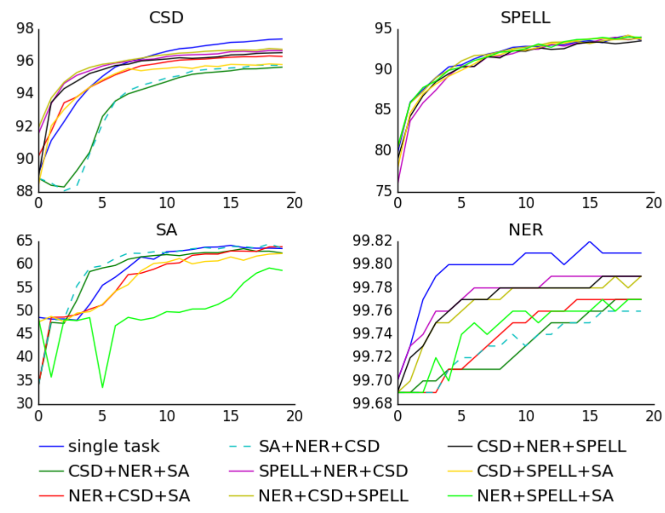

Figure 3: Accuracy (%) of jointly learning 3 tasks for 20 epochs with varying task order.

The results suggest that the order of the tasks has a different effect on the different tasks, for the first few epochs. For instance, while training SA+NER+CSD has a negative effect on both CSD and NER, it has a positive effect on SA (outperforms even SA trained separately). Likewise for CSD-NER-SA but at different extent. NER+CSD+SA has a negative effect on SA and NER overall, but it has a positive effect on CSD at the beginning. This suggests that the order of the tasks affects strongly the first epoch.
The same observation could be applied when jointly learning the 4 tasks as shown in Figure 4. In more details, jointly learning the 4 tasks in NER+CSD+SPELL+SA and SPELL+CSD+NER+SA orders improves SPELL, where the task achieves its best performance. While the same task orders have no positive effect on NER, they do boost the performance of CSD and SA in the beginning but eventually they cause the overall performance to level faster.

5.3. Context of words for SPELL

So far SPELL is trained at a sequence level (as a sequence-to-sequence). In order to measure the effect of the word context we train the same model architecture at a token level, and we refer to it as SPELL-token in Figure 5. The choice of NER+CSD+SPELL+SA order is based on the aforementioned results in Figure 4 where the selected task order performs the best for SPELL (in red). The results indicate clearly that context does matter for SPELL when

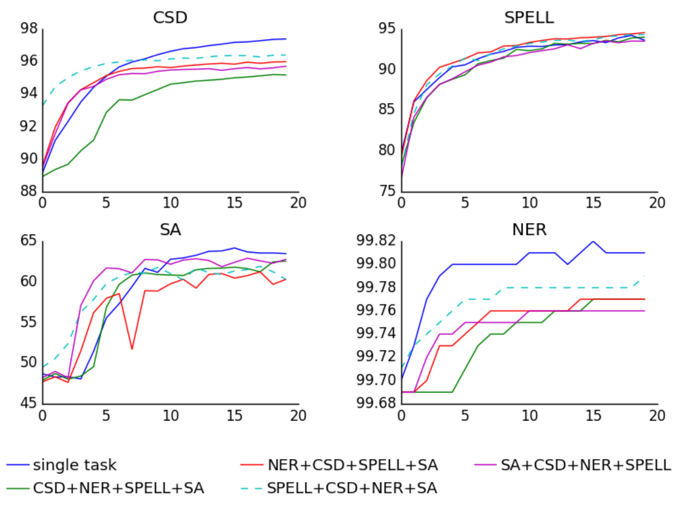

Figure 4: Accuracy (%) of jointly learning 4 tasks for 20 epochs with varying task order.

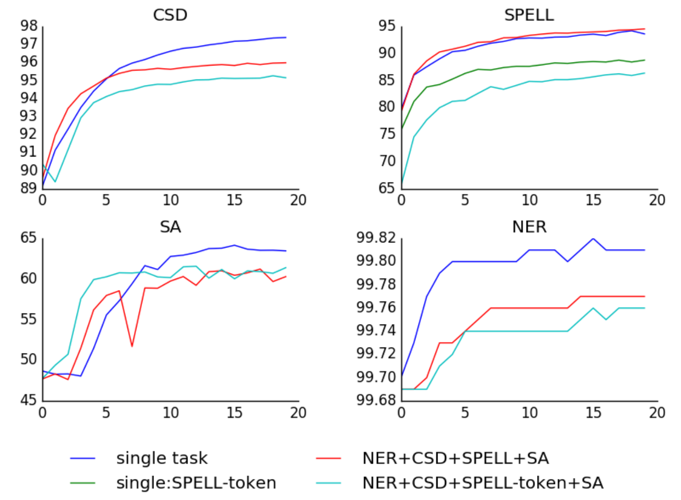

Figure 5: Accuracy (%) of jointly learning 4 tasks with(out) word context for SPELL.

trained separately and for CSD and NER tasks when trained jointly with SPELL and SA. Surprisingly, SPELL (with context) has a positive effect on SA only for the first 6 epochs then the effect is reversed. SPELL-token has an even more positive effect on SA before epoch 8. This suggests that either SA and SPELL datasets could include more ambiguity compared to other datasets, or that the noise of the two datasets hinders learning the tasks jointly.

5.4. Size of SPELL training data

As mentioned earlier, in this experiment we only vary the size of the training set for SPELL. We try 10k, 50k, 100k and all (185k)) and keep the rest unchanged to investigate whether this has any impact on jointly learning the tasks. We use the same task order, namely NER+CSD+SPELL+SA as motivated earlier, and we refer to it as multi-task in Figure 6.
In single task learning, the learning curves of SPELL in Figure 6 indicate that the performance of the task improves quickly with more data (by comparing the performances of

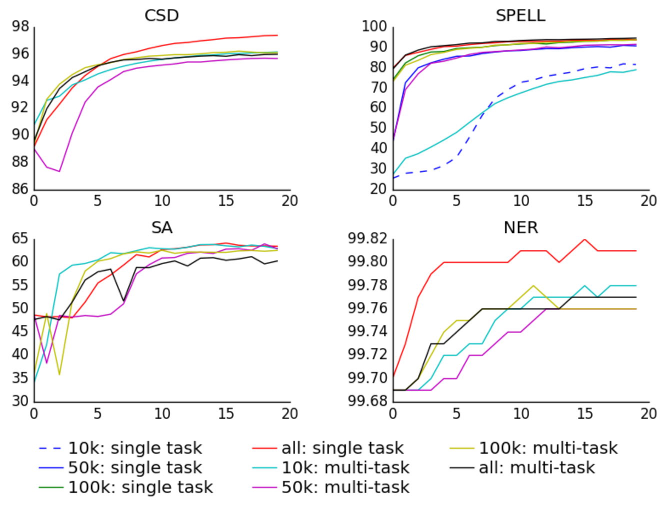

Figure 6: Accuracy (%) of jointly learning 4 tasks with varying SPELL training size. Single task: learning each task separately (baselines). Multi-task: jointly learning the tasks in NER+CSD+SPELL+SA order. All: train on all training sets as described in Section 5.

100k to 10k and 50k training samples). However, the performance levels with 100k samples, even though it takes a few more epochs to reach the performance than when using all training data. Towards the end the two lines are almost superposed. One possible explanation is that most representative data is already covered in 100k samples (the model has already seen enough data to achieve its maximum performance).

In multi-task learning, the same trend of single task learning is observed for SPELL with a small gain in the performance in the beginning when multi-tasking. Interestingly, as the amount of data increases, the gain of multi-tasking diminishes. For CSD, increasing the training size of SPELL from 10k to 50k has a negative effect, but increasing the size to 100k has boosted the performance of CSD especially in the beginning. The same thing is observed for NER and SA. One possible explanation could be that the datasets of 50k or less are too small and subject to random noise.

The best gain of multi-task for CSD is achieved when trained with only 100k of SPELL. NER and SA, exceeding even single task, benefits the most when trained with only 10k of SPELL. SPELL nevertheless follows the "more data better performance" hypothesis.

5.5. Data augmentation

We replicate the same experiment as in Section 5.3, but instead of comparing the performance of SPELL-token and SPELL separately, we augment the SPELL training data by combining both (token and sequence as input). This allows us to optimise the gain, if any, from the SPELL data.

Results in Figure 7 show that multi-task with the augmented data has arguably very little effect on SPELL compared to the single task in the same setting (the two lines are nearly superposed). However, data augmentation boosts the performance of SPELL compared to non-augmented data and even achieves its best performance. This rejects again hypothesis (2) in Section 5.1 because the performance of

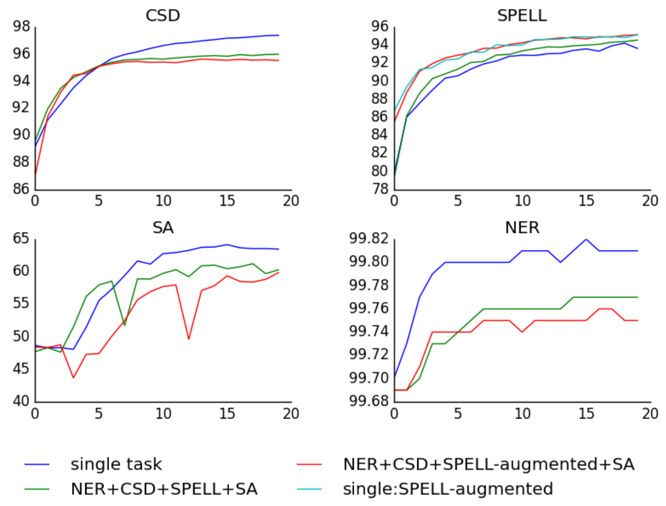

Figure 7: Accuracy (%) of jointly learning 4 tasks with data augmentation for SPELL. Augmented: using token + sequence as input to SPELL.

SPELL keeps increasing with more data.

On the one hand, augmenting SPELL data has a notable positive effect on SPELL when jointly trained with the other tasks compared to the same setting with non-augmented data (comparing green and red lines). On the other hand, in terms of effect on the other tasks, while augmenting SPELL data has a negative impact on SA, it offers a small benefit for CSD and NER at the very beginning (before epoch 6), but it is outperformed by the non-augmented data after that.

6. Conclusions and Future Work

We have examined the effect of jointly learning 4 tasks, which are neither morpho-syntactic nor semantic tagging, for noisy user-generated Algerian texts. The main findings of our empirical investigation, which includes a variety of experiments, could be summarised in the following points. (1) Tasks have different impacts on each other when learned jointly. (2) In multi-task learning notable gains are achieved for some tasks when trained jointly with specific tasks. Other tasks benefit from jointly learning them with some other tasks but the gain is only during the first few epochs, especially for tasks with little training data (CSD, NER and SA comparably to SPELL). Training for more epochs degraded their performance compared to learning them separately which is likely caused by the noisiness and sparsity of the data.

This means that it is hard to say whether multi-tasking is useful or not without mentioning several factors such as the tasks themselves, their order, the size of their datasets. (3) Word context for SPELL does matter for the task itself (single task) and for the tasks it is jointly trained with. (4) More SPELL training data does not necessary yield better results neither for the task itself (single task) nor for the tasks it is jointly learned with. In fact, performance is levelling at a certain point, in our case 10k samples for SA and NER, 100k for CSD, confirming this hypothesis. (5) Combining token and sequence level SPELL (augmented) is more

beneficial for the task itself (single task) with no gain for multi-task at the convergence point.
In the future, we will examine hypothesis (3) using sequential transfer learning, for instance by running SPELL on all datasets and compare their performances to the non spell corrected and normalised ones. Furthermore, we plan to explore the idea of curriculum learning (Elman, 1993; Hacohen and Weinshall, 2019) on both tasks and individual classes for each task by introducing the tasks or the classes in increasing order of difficulty.

7. Acknowledgements

The research reported in this paper was supported by a grant from the Swedish Research Council (VR project 2014-39) for the establishment of the Centre for Linguistic Theory and Studies in Probability (CLASP) at the University of Gothenburg.

8. Bibliographical References

Adouane, W. and Dobnik, S. (2017). Identification of Languages in Algerian Arabic Multilingual Documents. In *Proceedings of the Third Arabic Natural Language Processing Workshop*, pages 1–8, Valencia, Spain, April. Association for Computational Linguistics (ACL).

Adouane, W., Bernardy, J.-P., and Dobnik, S. (2018). Improving Neural Network Performance by Injecting Background Knowledge: Detecting Code-switching and Borrowing in Algerian texts. In *Proceedings of the Third Workshop on Computational Approaches to Linguistic Code-Switching*, pages 20–28, Melbourne, Australia, July. Association for Computational Linguistics (ACL).

Adouane, W., Bernardy, J.-P., and Dobnik, S. (2019). Normalising Non-standardised Orthography in Algerian Code-switched User-generated Data. In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*, pages 131–140, Hong Kong, China, November. Association for Computational Linguistics.

Adouane, W., Touileb, S., and Bernardy, J.-P. (2020). Identifying Sentiments in Algerian Code-switched User-generated Comments. In *Proceedings of the 12th Edition of Language Resources and Evaluation Conference (LREC)–To appear*, Marseille, France, May. European Language Resources Association (ELRA).

Bingel, J. and Søgaard, A. (2017). Identifying Beneficial Task Relations for Multi-task Learning in Deep Neural Networks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 164–169, Valencia, Spain, April. Association for Computational Linguistics (ACL).

Bollmann, M., Søgaard, A., and Bingel, J. (2018). Multi-Task Learning for Historical Text Normalization: Size Matters. In *Proceedings of the Workshop on Deep Learning Approaches for Low-Resource NLP*, pages 19–24, Melbourne, Australia. Association for Computational Linguistics (ACL).

Caruana, R. (1997). Multitask Learning. *Machine Learning*, 28:41–75.

Changpinyo, S., Hu, H., and Sha, F. (2018). Multi-Task Learning for Sequence Tagging: An Empirical Study. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2965–2977, Santa Fe, New Mexico, USA, August. Association for Computational Linguistics (ACL).

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, oct. Association for Computational Linguistics (ACL).

Collobert, R. and Weston, J. (2008). A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 160–167, New York, NY, USA. ACM.

Doyle, G. (2014). Mapping Dialectal Variation by Querying Social Media. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 98–106. Association for Computational Linguistics (ACL).

Eisenstein, J. (2013). Phonological Factors in Social Media Writing. In *Proceedings of the Workshop on Language Analysis in Social Media*, pages 11–19, Atlanta.

Elfardy, H., Al-Badrashiny, M., and Diab, M. (2013). Code Switch Point Detection in Arabic. In Elisabeth Métais, et al., editors, *Natural Language Processing and Information Systems*, pages 412–416, Berlin, Heidelberg. Springer Berlin Heidelberg.

Elman, J. (1993). Learning and Development in Neural Networks: the Importance of Starting Small. *Cognition*, 48:71–99, 08.

Hacohen, G. and Weinshall, D. (2019). On The Power of Curriculum Learning in Training Deep Networks. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pages 2535–2544.

Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Hovy, D. and Spruit, S. L. (2016). The Social Impact of Natural Language Processing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 591–598, Berlin, Germany, August. Association for Computational Linguistics (ACL).

Jørgensen, A., Hovy, D., and Søgaard, A. (2015). Challenges of Studying and Processing Dialects in Social Media. In *Proceedings of the Workshop on Noisy User-generated Text*, pages 9–18. Association for Computational Linguistics (ACL).

Luong, T., Le, Q. V., Sutskever, I., Vinyals, O., and Kaiser, L. (2016). Multi-task Sequence to Sequence Learning. In *International Conference on Learning Representations (ICLR)*.

Major, R. C. (2002). The Bilingualism Reader. Li Wei (ed.). London: Routledge, 2000. *Studies in Second Language Acquisition*, 24:491 – 493.

Martínez Alonso, H. and Plank, B. (2017). When is Multi-task Learning Effective? Semantic Sequence Prediction under Varying Data Conditions. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 44–53, Valencia, Spain, April. Association for Computational Linguistics (ACL).

Milroy, J. (2001). Language Ideologies and the Consequence of Standardization. *Journal of Sociolinguistics*, 5:530 – 555, 11.

Peng, N. and Dredze, M. (2017). Multi-task Domain Adaptation for Sequence Tagging. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 91–100, Vancouver, Canada, August. Association for Computational Linguistics (ACL).

Rickford, J. (1990). Dialects in Contact. *Language in Society - Oxford and New York: Basil Blackwell, 1986*, 19.

Samih, Y. and Maier, W. (2016). Detecting Code-switching in Moroccan Arabic. In *Proceedings of SocialNLP @ IJCAI-2016*.

Sayahi, L. (2014). *Diglossia and Language Contact: Language Variation and Change in North Africa*. Cambridge Approaches to Language Contact. Cambridge University Press.

Yin, W., Schütze, H., Xiang, B., and Zhou, B. (2015). ABCNN: Attention-Based Convolutional Neural Network for Modeling Sentence Pairs. *Transactions of the Association for Computational Linguistics*, 12.