Massive Choice, Ample Tasks (MaChAmp): A Toolkit for Multi-task Learning in NLP

Rob van der Goot* **Ahmet Üstün*** **Alan Ramponi*** **Barbara Plank**
IT University of Copenhagen University of Groningen University of Trento
Fondazione the Microsoft Research - University of Trento COSBI
robv@itu.dk, a.ustun@rug.nl, alan.ramponi@unitn.it, bapl@itu.dk

Abstract

Transfer learning, particularly approaches that combine multi-task learning with pre-trained contextualized embeddings and fine-tuning, have advanced the field of Natural Language Processing tremendously in recent years. In this paper we present MaChAmp, a toolkit for easy use of fine-tuning BERT-like models in multi-task settings. The benefits of MaChAmp are its flexible configuration options, and the support of a variety of NLP tasks in a uniform toolkit, from text classification to sequence labeling and dependency parsing. The code is available at: https://github.com/machamp-nlp/machamp (Version 0.1).

1 Introduction

Multi-task learning (MTL) (Caruana, 1993, 1997) has developed into a standard repertoire in NLP. It allows for learning tasks in parallel in neural networks (Caruana, 1993) while leveraging the benefits of sharing parameters. The shift—or the "tsunami" (Manning, 2015)—of deep learning in NLP has facilitated the wide-spread use of MTL since the seminal work by Collobert et al. (2011), which has lead to a multi-task learning "wave" (Ruder and Plank, 2018) in NLP. It has since been applied to a wide range of NLP tasks, developing into a viable alternative to classical pipeline approaches. This includes early adoption in Recurrent Neural Network type of models, e.g. (Lazaridou et al., 2015; Chrupała et al., 2015; Plank et al., 2016; Søgaard and Goldberg, 2016; Hashimoto et al., 2017), to the use of several unsupervised multi-task objectives to train BERT-like Language Models (Devlin et al., 2019) and ultimately, their combination in (low) supervised data regimes to fine-tune contextualized word embeddings with supervised objectives (Sanh et al., 2019).

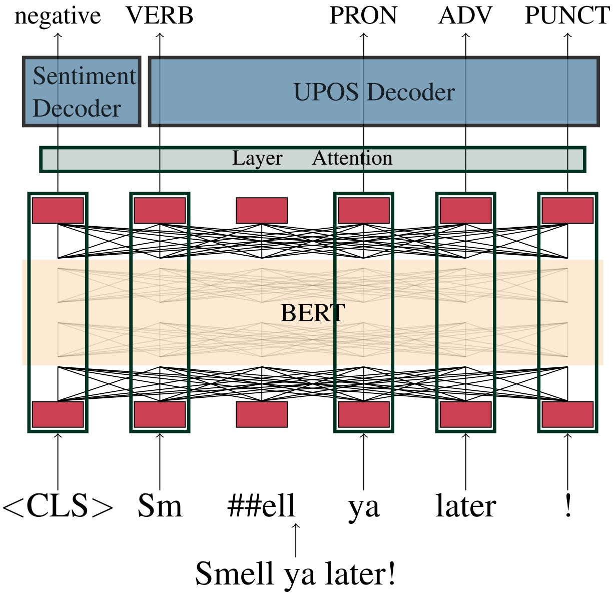

Figure 1: Overview of MaChAmp, when training jointly for sentiment analysis and POS tagging. A shared encoding representation and task-specific decoders are exploited to accomplish both tasks.

The key idea of language model pre-training and contextual embeddings (Howard and Ruder, 2018a; Peters et al., 2018; Devlin et al., 2019) is to pre-train rich representation on large quantities of monolingual or multilingual text data. Taking these representations as starting point has lead to enormous improvements across a wide variety of NLP problems. Effective models emerged for many languages and tasks (Hashimoto et al., 2017; Kondratyuk and Straka, 2019; Sanh et al., 2019; Hu et al., 2020). MTL comes in many flavours, based on the type of sharing, the weighting of losses, and the design and relations of tasks and layers. In general though, outperforming single-task settings remains a challenge (Martínez Alonso and Plank, 2017; Clark et al., 2019). For an overview of MTL in NLP we refer to (Ruder, 2017).

*Equal contribution

We introduce MaChAmp, a flexible toolkit for multi-task learning and fine-tuning of NLP problems. The main advantages of MaChAmp are:

- Ease of configuration, especially for multi-task setups;
- Support of a wide range of NLP tasks, from sequence labeling to dependency parsing and text classification;
- Support for the processing of multiple datasets at once;
- MaChAmp supports initialization with and finetuning of BERT embeddings (Devlin et al., 2019), which have shown to achieve state-of-the-art results for many NLP tasks.

For MaChAmp, we took a recent multilingual multi-task Universal Dependency parser (UDify) as starting point, which *in a single model* obtains competitive performance on 75 languages and all UD tasks (Kondratyuk and Straka, 2019). UDify however is targeted to Universal Dependencies (UD) parsing only. Consequently, their code and data handling logic was specifically designed and tailored for UD data. Kondratyuk and Straka (2019) used multilingual BERT[1] (mBERT) and fine-tuned its parameters for all UD tasks at hand (Nivre et al., 2016): part-of-speech tagging, lemmatization, morphological tagging, and dependency parsing.

We propose MaChAmp, which includes the functionality of UDify, yet supports many more data formats and tasks, and with its easy configuration, opens up for general use on more NLP tasks (see Section 2.2). Moreover, we plan to extend MaChAmp continuously and document the releases by updating this reference. The backbone of MaChAmp and UDify is AllenNLP (Gardner et al., 2018), a PyTorch-based (Paszke et al., 2019) Python library containing modules for a variety of deep learning methods and NLP tasks. It is designed to be modular, high-level and flexible.

2 Model

In this section we will discuss the model and its supported tasks.

2.1 Model overview

An overview of the model is shown in Figure 1. MaChAmp takes a pre-trained BERT model (Devlin et al., 2019) as initial encoder, and fine-tunes its layers by applying an inverse square root learning rate decay with linear warm-up (Howard and Ruder, 2018b), according to a given set of downstream tasks. For the task-specific predictions (i.e., decoding), each task has its own decoder, which is trained only for the corresponding task. By default all task-specific decoders are placed at the top-layer of the encoder (BERT); the configuration is though flexible (see Section 4.1). To make sure the BERT layers are used optimally, a layer attention over all layers is used for each decoder.

All the input is converted to WordPieces (Wu et al., 2016), an extension of Byte Pair Encodings (BPE) by Sennrich et al. (2016). For word-level tasks, the first WordPiece item of a word is used for the prediction. For sentence level tasks, the pooled output of the `[CLS]` token is used.

When multiple datasets are used for training, they are first separately split into batches (so each batch only contains instances from one dataset), which are then added together and shuffled before training. This means that small datasets will be underrepresented, which can be overcome by enabling proportional sampling (Section 4.2). During decoding, the loss function is only activated for tasks which are present in the current batch. By default all tasks have an equal weight in the loss function. The weight can be tuned, see Section 4.

2.2 Supported tasks

In the following, we describe the tasks MaChAmp supports. We distinguish two main types of tasks, one where the annotation is done on the word level (i.e. word-level tasks), and one where longer utterances of text are annotated with labels. For the latter, MaChAmp currently only supports sentence classification.

Sequence labeling This is to support classical token-level sequence prediction tasks, like part-of-speech tagging. Currently, MaChAmp uses greedy decoding with a softmax output layer from the hidden BERT WordPiece representation, similar as in (Kondratyuk and Straka, 2019).

String2string This is an extension to sequence labeling, which learns a conversion for each input word to its label. Instead of predicting the labels

[1] `https://github.com/google-research/bert/blob/master/multilingual.md`

directly, the model can now learn to predict the conversion. This strategy is commonly used for lemmatization (Chrupała, 2006; Kondratyuk and Straka, 2019), where it greatly reduces the label vocabulary. We use the transformation algorithm from UDPipe-Future (Straka, 2018), which was also used by Kondratyuk and Straka (2019).

Dependency parsing As in UDify (Kondratyuk and Straka, 2019), MaChAmp implements the deep biaffine parser (Dozat and Manning, 2017) using the Chu-Liu/Edmonds algorithm (Chu, 1965; Edmonds, 1967) for decoding the final tree. The default evaluation metric is LAS over all tokens (as opposed to accuracy for the other task types).

Text classification For text classification, MaChAmp predicts a label for every text by using the pooled `[CLS]` output of BERT. Pooling is performed by a feed-forward layer with a tanh activation which is trained with Next Sentence Prediction objective (Devlin et al., 2019) during pre-training. For tasks which model a relation between multiple sentences (e.g. RTE), a special `[SEP]` token is automatically inserted, so that the model can take this into account.

3 Usage

To use MaChAmp, one needs a configuration file, input data and a command to start the training or prediction. In this section we will describe each of these requirements.

3.1 Configuration

The model requires two configuration files, one that specifies the datasets and tasks, and one for the hyperparameters.[2] In the following subsections, we will describe the most useful options for both configurations.

A simple example of a dataset configuration file is shown in Figure 2. On the first level, the dataset names are specified (i.e., "UD" and "RTE"), which should be unique identifiers. Each of these datasets needs at least a `train_data_path`, a `validation_data_path`, a word index (i.e., `word_idx`) or sentence indices (i.e., `sent_idxs`), and a list of `tasks`. The word index specifies on which column the input words are to be read (see Section 3.2). For each of the defined tasks, the user is required to define the `task_type` (Section 2.2), and the column index from which to read the labels (i.e., `column_idx`). More options that can be passed on the task level are discussed in Section 4.1.

```
{
  "UD": {
    "train_data_path": "data/ewt.train",
    "validation_data_path": "data/ewt.dev",
    "word_idx": 1,
    "tasks": {
      "lemma": {
        "task_type": "string2string",
        "column_idx": 2
      },
      "upos": {
        "task_type": "seq",
        "column_idx": 3
      }
    }
  }
  "RTE": {
    "train_data_path": "data/RTE.train",
    "validation_data_path": "data/RTE.dev",
    "sent_idxs": [0,1],
    "tasks": {
      "rte": {
        "task_type": "classification",
        "column_idx": 2
      }
    }
  }
}
```

Figure 2: Example dataset configuration file, to predict UPOS, lemmas, and textual entailment simultaneously.

3.2 Data format

MaChAmp supports two types of data formats, which correspond to the level of annotation (Section 2.2). For word-level tasks, we will use the term "word-level file format", whereas for sentence-level task, we will use "sentence-level file format"

The word-level file format is similar to the CoNLLU format (Nivre, 2015) introduced for Universal Dependencies. It assumes one word per line, with each annotation layer following each word separated by a tab character (Figure 3a). Sentences are delimited by an empty line. Comments are lines on top of the sentence which have a different number of columns with respect to token lines.[3] It should be noted that for dependency parsing, it assumes the relation label to be on the `column_idx` and the head index on the following column.

[2] For the hyperparameters configuration a default option is already specified (`configs/params.json`), which should give reasonably high performance for most tasks.

[3] We do not identify comments based on lines starting with a '#', because datasets might have words in the first column that can start with a '#'.

```
1    smell    VERB
2    ya   PRON
3    later    ADV
4    !    PUNCT
```

(a) Example of a word-level file format, where word_idx should be 1, and task_idx 2.

```
smell ya later !    negative
```

(b) Example of a phrase-level file format, where sent_idxs should be [0] and task_idx 1.

Figure 3: Examples of data file formats.

The sentence-level file format for sentence classification is very similar (Figure 3b), except that there can be multiple inputs. In contrast to word_idx, a list of sent_idxs are defined to enable modeling the relation between any arbitrary number of them.

3.3 Training

Given the setup illustrated in the previous sections, a model can be trained using the following command. It assumes the configuration (Figure 2) called configs/upos-lemma-rte.json.

```
python3 train.py --parameters_config \
 configs/params.json --dataset_config \
 configs/upos-lemma-rte.json
```

As is common in AllenNLP (Gardner et al., 2018), by default the model and the logs will be written to logs/<JSONNAME>/<DATE>. The name of the directory can be set manually by providing --name <NAME>. Furthermore, --device can be used to specify which GPU to use (-1 for CPU is the default).

3.4 Inference

Prediction on new data can then be done using the following command:

```
python3 predict.py \
 logs/<NAME>/<DATE>/model.tar.gz \
 <INPUT FILE> <OUTPUT FILE>
```

It requires the path to the best model serialized during training stored as model.tar.gz in the log directory as specified above.

4 Options

For the full list of tuning hyperparameters, see the default configs/params.json in the repository (and Section 5.2). In this section we first discuss the parameters which can be defined for tasks individually, and then describe how to change the pre-trained embeddings.

4.1 Dataset configuration

The settings described in this section can only be set on the task level.

Metric Can be used to specify the evaluation metric. If not set, it defaults to accuracy, except when task_type is set to dependency, then LAS, as defined by AllenNLP (Gardner et al., 2018), is used. Possible metrics are: 'acc', 'LAS', 'micro-f1', 'macro-f1' and 'span_f1' (for span-based sequence labeling).

Layer Defines which layers are used for predicting the task. The model uses up to the specified layer (if it set to 8, it uses all layers from 1 to 8). As explained in Section 2, by default the model uses layer attention to mix the information from the specified layers.

Loss weights In multi-task settings, not all tasks might be equally important, or some tasks might just be harder to learn, and therefore should gain more weight during training. This can be done by setting the loss_weight parameter on the task level (by default the value is 1.0 for all tasks).

Adaptive This enables the adaptive softmax loss function (Grave et al., 2017). This loss function groups the labels into several clusters based on their frequency. By first focusing on the more frequent labels the model can gain both in efficiency and in performance. This should mostly be beneficial for tasks with imbalanced label spaces.

Following UDify, the adaptive softmax loss is set to True by default (with cutoff values 8 and 15), as it generally improves performance. However, when the label vocabulary setting for a task is lower than 8, the cutoff value does not apply and adaptive softmax is not active.

4.2 Hyperparameter configuration

Whereas most of the hyperparameters can simply be changed from the default parameters (configs/params.json) in the repository, we would like to highlight two settings.

BERT model The path to pre-trained BERT can be set in the params.json file. Specifically the pretrained_model value in

Parameter	Value	Range
Optimizer	Adam	
β_1, β_2	0.9,0.99	
Weight decay	0.01	
Label smoothing	0.03	
Dropout	0.5	0.3, 0.5, 0.7
BERT dropout	0.1	0.1, 0.2
Mask probability	0.1	0.1, 0.2, 0.3
Layer dropout	0.1	
Batch size	32	16, 32, 64
Epochs	80	
Patience	5	
Base learning rate	.001	.0001, .001, .01
BERT learning rate	$5e^{-5}$	
Warmup rate	1/80	
Gradient clipping	5.0	
Dep. tag dimension	256	
Dep. arc dimension	768	

Table 1: Final parameter settings, incl. tested ranges.

the `datasetreader/bert/` section. The model expects the embeddings to be in pytorch format, which can be obtained by the `pytorch_transformers` command.

Proportional sampling To avoid larger datasets from overwelming the model, proportional sampling can be enabled (`iteration/proportional_sampling`). In previous work this has shown to be reach high performance when modeling multiple tasks hierarchically (Sanh et al., 2019). When enabled, the model will first pick a random task, and then pick a random batch from that task. In other words, all datasets will have a roughly equal amount of batches; smaller datasets will be up-scaled and larger datasets will be downscaled (the number of batches per batch remains the same).[4] It should be noted that for specific tasks, more involved strategies have been devised (Wang et al., 2020; Stickland and Murray, 2019).

5 Experiments

In this section we describe the procedure how we determined robust default parameters for MACHAMP; note that the goal is not to achieve a new state-of-the-art, but generally to reach on-par performance for multiple tasks, while reaching one robust setting of hyperparameters. To this end, we will describe the datasets and hyperparameters used for tuning, and the obtained results in single-task and multi-task setups.

5.1 Datasets

We report performance over three benchmarks. They were selected to cover a range of NLP tasks, from syntactic to semantic and inference-level tasks, spanning the diverse supported tasks and different dataset setups. Next we describe each dataset and all its tasks. For simplicity (and due to availability), we focus on English datasets only.

UD (EWT) The English Universal Dependencies (Silveira et al., 2014) data derived from the English Web Treebank. It is the English dataset which was the earliest part of UD English. It is the most commonly used dataset for UD English dependency parsing. It contains 5 tasks: fine and coarse-grained POS tagging (XPOS and UPOS, respectively), lemmatization, morphological tagging and parsing. This dataset is an example where all tasks are annotated jointly for every instance.

GLUE The General Language Understanding Evaluation benchmark (Wang et al., 2018) has become the default benchmark for inference-type or semantics tasks, including entailment, paraphrasing and sentiment analysis. It consists of a collection of several datasets (Warstadt et al., 2019; Socher et al., 2013; Dolan and Brockett, 2005; Cer et al., 2017; Williams et al., 2018; Rajpurkar et al., 2018; Bentivogli et al., 2009; Levesque et al., 2012), and hence represents an example of a dataset with multiple disjointly labeled datasets. We use all datasets except for SST, which is a regression task, and WNLI, following previous work (Devlin et al., 2019).

PMB The Parallel Meaning Bank (Bos, 2015; Abzianidze et al., 2017) is a multilingual data collection for semantic processing. It includes a range of basic tasks, all of which form the basis to generate a Discourse Representation Structure (DRS) using Boxer (Bos, 2015). The base tasks, which we evaluate on here, include: CCG supertagging, semtagging, verbnet and wordnet tagging. We use the English part and PMB version 3.0.0. It should be noted that for the wordnet senses, we used the `string2string` task type. This reduced the vocabulary size from 4,443 to 1,804. This is arguably

[4] We also experimented with only up-scaling and only down-scaling, but found the first to be too efficient and the latter to lead to sub-optimal performance

	EWT v2.3					PMB v3.0				
Task	dep	feats	lemma	upos	xpos	lemma	semtag	supertag	verbnet	wordnet
Task type	dep	seq	s2s	seq	seq	s2s	seq	seq	seq	s2s
Train size			205k					43k		
MaChAmp$_{(ST)}$	**89.90**	**97.18**	**98.21**	**97.01**	96.64	**97.52**	**98.32**	94.87	94.37	89.15
MaChAmp$_{(MT)}$	89.61	97.15	97.79	**97.01**	**96.79**	97.33	98.23	**94.91**	**94.54**	**89.32**
UDify	89.67	97.15	97.80	96.90	–	–	–	–	–	–

	GLUE								
Task	cola	mnli	mnli-mis	mrpc	qnli	qqp	rte	snli	sst-2
Task type	c	c	c	c	c	c	c	c	c
Train size	8.5k	392k	392k	3.6k	108k	363k	2.5k	549k	67k
MaChAmp$_{(ST)}$	**78.04**	81.99	82.15	**86.03**	**88.31**	**89.75**	72.20	89.58	**90.71**
MaChAmp$_{(MT)}$	72.20	82.35	**82.80**	82.11	86.58	89.27	**73.65**	89.61	90.25
MaChAmp$_{(MT+P)}$	76.03	80.40	80.55	84.31	87.26	87.40	73.29	87.41	90.14
BERT-base	–	**84.4**	**86.7**	–	–	–	–	**93.3**	–

Table 2: Performance of MaChAmp$_{(ST)}$ (single-task), MaChAmp$_{(MT)}$ (multi-task), and MaChAmp$_{(MT+P)}$ (multi-task+proportional sampling) on the dev sets. For all tasks, accuracy is used as metric, except for dependency parsing where LAS score is used. Training size is the number of annotated instances (words in case of EWT and PMB, sentences for GLUE). Results reported per dataset are from UDify (Kondratyuk and Straka, 2019) and BERT-base (Devlin et al., 2019). The task types are; dep: Dependency parsing; seq: Sequence labeling; c: text classification; s2s: String2string.

a strange task to tackle, because we predict word-senses, without knowing which senses exist, which is why the performance is lower compared to the other tasks (Table 2).

5.2 Hyperparameter tuning

Because UDify (Kondratyuk and Straka, 2019) was focused on training on many UD parsing datasets and languages simultaneously, its hyperparameters were tuned towards massive data sizes. We compared a range of hyperparameter settings for our three setups by using grid search,[5] and used the parameter settings that reached the highest rank (averaged over the three sets). In Table 1 we report the best hyperparameters across all datasets (these are the default of the toolkit), and the range of parameters which we evaluated. Patience and model selection is based on the sum of all the evaluation metrics of all tasks.

5.3 Results

The final performance for all datasets and tasks on the development data are reported in Table 2. For each dataset we ran a multi-task model (MaChAmp$_{(MT)}$), performing all tasks jointly (for GLUE, we train on all datasets jointly). We compare this to single-task model (MaChAmp$_{(ST)}$), where we train a separate model for each task. .

First, we see that MaChAmp obtains state-of-the-art performance for EWT for which we can directly compare on all tasks except fine-grained POS tagging. In more detail, the results of UDify (Kondratyuk and Straka, 2019) are in a similar range for all tasks; the largest difference and improvement for MaChAmp is on dependency parsing, which is probably due to parameter tuning. The multi-task model works well and for both EWT and PMB performs similar to the single-task models; on some of the task MaChAmp even slightly outperforms them.

However, on GLUE performance of the multi-task model lacks behind on 6 out of the 9 tasks. This is probably due to the immense size and disparity of the datasets. Proportional sampling helps to train the model more quickly and especially for the smaller datasets results in better accuracies. Comparing to the original results from BERT (Devlin et al., 2019), we see that their scores are higher. This is probably because of the differences in the setup: they used BERT-base, we use BERT-multilingual, also they only trained for 3

[5] We capped the dataset sizes to a maximum of 20,000 sentences for efficiency reasons.

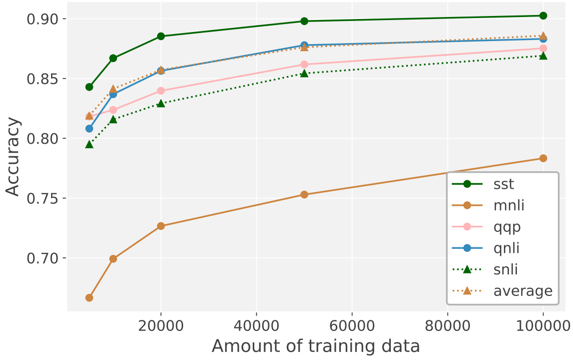

Figure 4: Learning curves for MACHAMP$_{(ST)}$ on the dev sets for the large GLUE training datasets.

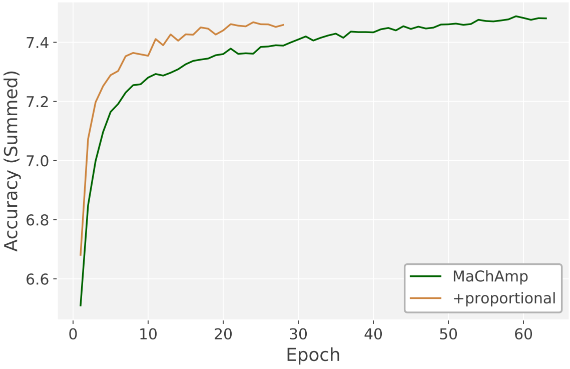

Figure 5: Results of MaChAmp with and without proportional sampling.

epochs which only makes sense for large datasets. Furthermore, they tuned the learning rate per task, whereas we were mainly looking for a robust setting over multiple datasets.

Learning Curve As training a joint MTL model on GLUE is computationally expensive (it takes several days, particularly when SNLI is included with over half a million training instances), we trained single-task models and examine the effect of increasing data size per GLUE task. We focus on the larger GLUE tasks which have more than 100k training instances. The learning curves shown in Figure 4 show that all tasks benefit from more GLUE training data, even beyond 50k instances accuracy keeps increasing, except for SST which starts to flatten out. MNLI is the task with the steepest learning curve and the lowest accuracy. This shows that the GLUE tasks remain challenging and more data clearly helps most of the tasks.

Proportional Sampling We examine the effect of proportional sampling (Section 4.1) only for the GLUE benchmark, as it contains datasets of varying sizes. The performances per epoch are plotted in Figure 5. The plot clearly shows the advantages of using proportional sampling; higher performance is obtained with fewer epochs. These results also show that there is more potential to be gained; perhaps using a dynamic ratio we can benefit from both types of training (also reflected in Table 2, where both models show very different strengths).[6]

6 Conclusions

We introduced MACHAMP, a flexible toolkit for BERT-based multi-task learning, and evaluated it on three multi-task benchmarks. Performance is on-par to previous state-of-art models, and even higher for some low-resource settings. The source code is freely available.

Acknowledgments

This research was supported by an Amazon Research Award, an STSM in the Multi3Generation COST action (CA18231), a visit supported by COSBI, grant 9063-00077B (Danmarks Frie Forskningsfond), and Nvidia corporation for sponsoring Titan GPUs. We thank the NLPL laboratory and the HPC team at ITU for the computational resources used in this work.

References

Lasha Abzianidze, Johannes Bjerva, Kilian Evang, Hessel Haagsma, Rik van Noord, Pierre Ludmann, Duc-Duy Nguyen, and Johan Bos. 2017. The parallel meaning bank: Towards a multilingual corpus of translations annotated with compositional meaning representations. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 242–247, Valencia, Spain. Association for Computational Linguistics.

Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. 2009. The fifth pascal recognizing textual entailment challenge. In *TAC*.

Johan Bos. 2015. Open-domain semantic parsing with boxer. In *Proceedings of the 20th Nordic Conference of Computational Linguistics (NODALIDA 2015)*, pages 301–304, Vilnius, Lithuania. Linköping University Electronic Press, Sweden.

[6] It should be noted that the proportional sampling model could have been somewhat ”unlucky” with the sampling, and might have benefited from training longer. The other model had many points where it did not improve for 3-4 epochs

Rich Caruana. 1993. Multitask learning: A knowledge-based source of inductive bias. In *Proceedings of the Tenth International Conference on Machine Learning*.

Rich Caruana. 1997. Multitask learning. In *Learning to learn*, pages 95–133. Springer.

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada. Association for Computational Linguistics.

Grzegorz Chrupała, Ákos Kádár, and Afra Alishahi. 2015. Learning language through pictures. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 112–118, Beijing, China. Association for Computational Linguistics.

Grzegorz Chrupała. 2006. Simple data-driven context-sensitive lemmatization. *SEPLN*.

Yoeng-Jin Chu. 1965. On the shortest arborescence of a directed graph. *Scientia Sinica*, 14:1396–1400.

Kevin Clark, Minh-Thang Luong, Urvashi Khandelwal, Christopher D. Manning, and Quoc V. Le. 2019. BAM! born-again multi-task networks for natural language understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5931–5937, Florence, Italy. Association for Computational Linguistics.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

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, Minneapolis, Minnesota. Association for Computational Linguistics.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Timothy Dozat and Christopher D Manning. 2017. Deep biaffine attention for neural dependency parsing. *Proceedings of 5th International Conference on Learning Representations (ICLR 2017)*.

Jack Edmonds. 1967. Optimum branchings. *Journal of Research of the national Bureau of Standards B*, 71(4):233–240.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. AllenNLP: A deep semantic natural language processing platform. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 1–6, Melbourne, Australia. Association for Computational Linguistics.

Edouard Grave, Armand Joulin, Moustapha Cissé, Hervé Jégou, et al. 2017. Efficient softmax approximation for gpus. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1302–1310.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 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, Copenhagen, Denmark. Association for Computational Linguistics.

Jeremy Howard and Sebastian Ruder. 2018a. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia. Association for Computational Linguistics.

Jeremy Howard and Sebastian Ruder. 2018b. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339.

Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and Melvin Johnson. 2020. Xtreme: A massively multilingual multi-task benchmark for evaluating cross-lingual generalization. *arXiv preprint arXiv:2003.11080*.

Dan Kondratyuk and Milan Straka. 2019. 75 languages, 1 model: Parsing universal dependencies universally. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2779–2795, Hong Kong, China. Association for Computational Linguistics.

Angeliki Lazaridou, Nghia The Pham, and Marco Baroni. 2015. Combining language and vision with a multimodal skip-gram model. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 153–163, Denver, Colorado. Association for Computational Linguistics.

Hector Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning*.

Christopher D Manning. 2015. Computational linguistics and deep learning. *Computational Linguistics*, 41(4):701–707.

Héctor Martínez Alonso and Barbara Plank. 2017. When is multitask 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*, Valencia, Spain. Association for Computational Linguistics.

Joakim Nivre. 2015. Towards a universal grammar for natural language processing. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 3–16. Springer.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1659–1666, Portorož, Slovenia. European Language Resources Association (ELRA).

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8026–8037. Curran Associates, Inc.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 412–418, Berlin, Germany. Association for Computational Linguistics.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia. Association for Computational Linguistics.

Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*.

Sebastian Ruder and Barbara Plank. 2018. Strong baselines for neural semi-supervised learning under domain shift. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1044–1054, Melbourne, Australia. Association for Computational Linguistics.

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.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Natalia Silveira, Timothy Dozat, Marie-Catherine de Marneffe, Samuel Bowman, Miriam Connor, John Bauer, and Christopher D. Manning. 2014. A gold standard dependency corpus for English. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

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)*, pages 231–235.

Asa Cooper Stickland and Iain Murray. 2019. Bert and pals: Projected attention layers for efficient adaptation in multi-task learning. In *ICML*.

Milan Straka. 2018. UDPipe 2.0 prototype at CoNLL 2018 UD shared task. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 197–207, Brussels, Belgium. Association for Computational Linguistics.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Xinyi Wang, Yulia Tsvetkov, and Graham Neubig. 2020. Balancing training for multilingual neural machine translation. In *Annual Conference of the Association for Computational Linguistics (ACL)*.

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.