Assignment 1: Cross-domain Adaptation

During these assignment we will evaluate a variety of adaptation approaches for a variety of models. We will use the task of sentiment analysis for these experiments. For the assignments we have provided two datasets; the Stanford Sentiment Treebank (SST) (Socher et al., 2013) and data from SemEval-2013 Task 2: Sentiment Analysis in Twitter (Nakov et al., 2013). Both of these datasets have been pre-processed to be in a similar format. The data from the SST dataset is relatively "clean" data, and is taken from review websites. The data from SemEval2013-2 is taken from Twitter, and can be expected to be more "noisy".

In the end, you should be able to fill out the following table (grey cells can be skipped), and also discuss what we can learn from these results:

Model	Reviews	Twitter	Danish
Logistic Regression			▨
Logistic Regression + characters			▨
Logistic Regression + normalization			▨
Logistic Regression + noisy train			▨
LSTM			▨
LSTM + word2vec			▨
LSTM + multi-task			▨
BERT			
BERT + domain retraining			▨
BERT + multi-lingual	▨	▨	

Table 1: All final results can be collected in a table like this; they should be accuracies for each model and dataset.

For the assignments, we ask you to create a PDF or a Python notebook with your answers as well as findings. For each assignment that starts with "Approach", you are expected to fill a row in the table and write down concisely what we can conclude based on these results. For the other assignments, there are specific questions. It is also allowed to hand in only the answers to a subset of the assignments.

You can re-use this table definition if you use LaTeX:

```
\usepackage{booktabs}

\begin{tabular}{l r r r}
\toprule
Model                               & Reviews & Twitter & Danish \\
\midrule
Logistic Regression                 & & & \cellcolor{gray!25}\\
Logistic Regression + characters    & & & \cellcolor{gray!25}\\
Logistic Regression + normalization & & & \cellcolor{gray!25}\\
Logistic Regression + noisy train   & & & \cellcolor{gray!25}\\
LSTM                                & & & \cellcolor{gray!25}\\
LSTM + word2vec                     & & & \cellcolor{gray!25}\\
LSTM + multi-task                   & & & \cellcolor{gray!25}\\
BERT                                & & & \\
BERT + domain retraining            & & & \cellcolor{gray!25}\\
BERT + multi-lingual                & \cellcolor{gray!25} & \cellcolor{gray!25} & \\
\bottomrule
\end{tabular}
```

Week 1: Logistic Regression

All the referenced files are available on LearnIt in the archive `assignment1.tar.gz`. The goal of this assignment is to:

- Get (re-)acquainted with classification problems in NLP.
- Learn to implement character-level features.
- Evaluate and analyze the result of an NLP experiment.

1 Baseline

(a) Inspect the development datasets of both domains. Are there any systematic differences observable in the input texts?

(b) Inspect the provided code in `code/1.logres.py`. What is used as input features for the classification?

(c) Run the baseline provided in `code/1.logres.py`, train it on `sst.train` and evaluate it on `sst.dev` and `semeval2013.dev` using accuracy. How large is the performance difference between the in-domain review data and the Twitter data?

2 Approach 1: Exploit Character Level Information

It is well known that unknown words are problematic for NLP systems. One way to circumvent this, is to also incorporate character level information. Adapt the classifier so that it uses character 3-6 grams and word 1-2 grams. To do this you need to create two vectorizers and combine the results into a single feature space. You can use the SKLearn FeatureUnion (`https://scikit-learn.org/stable/modules/generated/sklearn.pipeline.FeatureUnion.html`) for this.

3 Approach 2: Convert Input Data

The normalized version (from MoNoise) of the development data is included in `semeval2013.dev.normed`. Now evaluate the performance of the model that uses both words and characters as features from assignment 1 on this data. Inspect the most common types of replacements done by the normalization to obtain more accurate findings (as opposed to only looking at the scores).

4 Approach 3: Convert Training Data

Create a version of the training data where for each character, one of 4 things can happen:

- It doesn't change: 91%
- A character is inserted after this character 3%
- The character is removed 3%
- The character is swapped with the previous character 3%

Train on the new training data, and evaluate on both dev sets.

5 Approach 4: Bonus

(a) Automatically create alternative versions of the training data in `sst.train`.

- You can take inspiration from the Ùfal paper (Samuel and Straka, 2021), or the GenERRate paper (Foster and Andersen, 2009).
- You can create as many variations of the training instances as you find useful. You can also include the original training data.
- Note that the goal is **not** to improve the model hyperparameters or add annotated training data.

(b) Train the model on the new training data and evaluate it on both dev sets.

Week 2: RNN and Multi-task Learning

The goal of this assignment is to:

- Get (re-)acquainted with BiLSTMs and PyTorch.
- Learn to adapt an NLP model to a new domain through language modeling.
- Implement a simple method for neural multi-task learning: adding a decoder head for each task.
- Compare performances of different settings, and think about how to interpret these results.

6 Baseline

The baseline model is an LSTM, familiarize yourself with the code in `code/2.bilstm.py`

(a) What does the input data look like after it has been returned from the `convert_data` function? (i.e. what is the shape of `train_features`?, what do the values represent?)

(b) What is the shape of the `logits` as returned by the forward function of the model? (not just the numbers, what do they represent?)

(c) In the final feedforward layer, the input dimension is set to `lstm_dim*2`. Why is the multiplication by 2 necessary?

(d) Evaluate a model trained on SST on both the in-domain data and the SemEval2013 dev data.

(e) How many unknown words are there in both dev datasets?

7 Approach 1: Using Word Embeddings for Adaptation

Initialization with pre-trained embeddings can be used to inject "knowledge" about a "language" into a model. In this assignment, we are going to use word embeddings trained with word2vec on Twitter data (`http://www.itu.dk/~robv/data/embeds/w2v-twitter.txt.gz`). More information about how these embeddings are trained can be found on: `https://robvanderg.github.io/modeling/twit-embeds/`

Load the embeddings, and use them to initalize the `self.word_embeddings` in the `Classificator model`. Some hints:

- You can use the `nn.Embedding.from_pretrained` function (`https://pytorch.org/docs/stable/generated/torch.nn.Embedding.html#torch.nn.Embedding.from_pretrained`)

- Note that this expects a 2-dimensional torch tensor, the first dimension matches the number of words (4,428,756), and the second dimension the size of the embeddings (100).
- The indices in the original matrix should match the word indices you use as input, so the `Vocabulary` class needs to be adapted. The easiest way to do this is probably to just create the word2idx and idx2word based on the embeddings file (note that the Unknown token is [UNK], and is at position 0 already)
- The embeddings file is in text format, the first line contains the dimensions, followed by a word with its representation per line:

```
4428756 100
[UNK] 0.004003 0.004419 -0.003830 -0.003278 0.001367 0.003021 0.000941 0.000211 -0.003604 0.002218
```

- For debugging as well as drawing conclusions it is useful to look at the number of unknown words again after incorporating the embeddings.

8 Approach 2: Multi-task Learning on Auxiliary Tasks

Sarcasm detection is relevant for sentiment analysis as it can invert the sentiment label. Hence, we will test the hypothesis whether we can use sarcasm detection as an auxiliary task to support our sentiment analysis system.

We will use data from the 2020 Sarcasm Detection Shared Task (Ghosh et al., 2020). We converted it to the same format as the sentiment data in `figlang.train` and `figlang.test`.

You can follow the following steps:

1. Add a feedforward classification layer to the existing model that can classify sarcasm. This is a simple method to implement multi-task learning, with the maximum amount of sharing, you can see a diagram of the resulting model in Figure 1.
2. Implement data reading for the sarcasm task; make sure the word embeddings (and word indices) are shared across tasks, but the label indices should not be shared!
3. Adapt the training procedure. You can do this by adding another for loop within each epoch in which you traverse through the batches of the second dataset. You can reuse the optimizer and the loss from the first task. Make sure that you use the right predictions and compare with the matching gold labels for each task.

Week 3: Transformers, BERT and Transfer Learning with Language Models

All referenced files are available on LearnIt in the archive `assignment-2.tar.gz`. The goal of this assignment is to:

- Learn how to use the HPC
- Be able to train, use, and evaluate a transformer-based language model for NLP tasks
- Evaluate the language-transfer abilities of multilingual language models

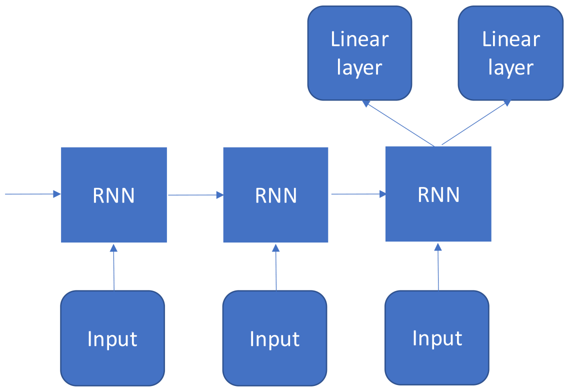

Figure 1: Simple multi-task model that performs two tasks simultaneously.

9 Baseline

In this assignment, we are going to fine-tune BERT models. Note that you need ≈8 GB of GPU-ram for this assignment (or a lot of patience). The ITU has an HPC that you can use for free (s(seeides for instructions on how to use it). We strongly recommend you to learn how to use it, as it allows you to do many more experiments for this course, and is also a valuable skill outside of ITU (the SLURM workload manager is very commonly used).

- Compare the code to the BiLSTM code and familiarize yourself with the differences.

Now train a sentiment analysis model with BERT (`bert-base-cased`) on the English SST data. You can use the code in `code/1.bert.py`.

10 Approach 1: Domain-adaptive Re-training

`cardiffnlp/twitter-xlm-roberta-base` (Barbieri et al., 2022) is a BERT-based model that is retrained on Twitter data. Train a sentiment model with these Twitter embeddings on the SST train data. Does it transfer better to the English Twitter data compared to the mBERT model?

11 Approach 2: Cross-lingual Transfer

In this assignment, we will look at cross-lingual transfer. We will compare the cross-lingual performance of a monolingual language model to the performance of a multilingual language model. We will use the English SST data for sentiment analysis for training and evaluating the models on Danish data (`https://github.com/alexandrainst/danlp/blob/master/docs/docs/datasets.md#twitter-sentiment-twitsent`).

(a) Train an English BERT model (`bert-base-cased` on huggingface) on the SST reviews data, evaluate it on the SST data as well as the Danish Twitter data. Is there a performance drop when going to the Danish data? How does the performance on the Danish data compare to the majority baseline?

(b) Train an mBERT model (`bert-base-multilingual-cased` on huggingface) model on the reviews data from SST, and evaluate it on the Danish development split. What is the performance? How does it compare to the English BERT model?

References

F. Barbieri, L. Espinosa Anke, and J. Camacho-Collados. XLM-T: Multilingual language models in twitter for sentiment analysis and beyond. In *Proceedings of the Language Resources and Evaluation Conference*, pages 258–266, Marseille, France, June 2022. European Language Resources Association. URL `http://www.lrec-conf.org/proceedings/lrec2022/pdf/2022.lrec-1.27.pdf`.

J. Foster and O. Andersen. GenERRate: Generating errors for use in grammatical error detection. In J. Tetreault, J. Burstein, and C. Leacock, editors, *Proceedings of the Fourth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 82–90, Boulder, Colorado, June 2009. Association for Computational Linguistics. URL `https://aclanthology.org/W09-2112/`.

D. Ghosh, A. Vajpayee, and S. Muresan. A report on the 2020 sarcasm detection shared task. In B. B. Klebanov, E. Shutova, P. Lichtenstein, S. Muresan, C. Wee, A. Feldman, and D. Ghosh, editors, *Proceedings of the Second Workshop on Figurative Language Processing*, pages 1–11, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.figlang-1.1. URL `https://aclanthology.org/2020.figlang-1.1/`.

P. Nakov, S. Rosenthal, Z. Kozareva, V. Stoyanov, A. Ritter, and T. Wilson. SemEval-2013 task 2: Sentiment analysis in Twitter. In S. Manandhar and D. Yuret, editors, *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 312–320, Atlanta, Georgia, USA, June 2013. Association for Computational Linguistics. URL `https://aclanthology.org/S13-2052/`.

D. Samuel and M. Straka. ÚFAL at MultiLexNorm 2021: Improving multilingual lexical normalization by fine-tuning ByT5. In W. Xu, A. Ritter, T. Baldwin, and A. Rahimi, editors, *Proceedings of the Seventh Workshop on Noisy User-generated Text (W-NUT 2021)*, pages 483–492, Online, Nov. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.wnut-1.54. URL `https://aclanthology.org/2021.wnut-1.54/`.

R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In D. Yarowsky, T. Baldwin, A. Korhonen, K. Livescu, and S. Bethard, editors, *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA, Oct. 2013. Association for Computational Linguistics. URL `https://aclanthology.org/D13-1170/`.