IN5550
III: Named entity recognition and pre-trained language models

Language Technology Group, IFI, UiO

Deadline: 23:59, April 12 2021

1 Goals

- Learn how to perform basic NLP operations using pre-trained contextualized models.
- Acquire experience with modern transformer-based architectures (BERT, etc)
- Apply pre-trained neural language models to a sequence tagging task in Norwegian

2 Recommended Reading

1. **Neural network methods for natural language processing.**, Goldberg, Y., 2017[1]
2. **Speech and Language Processing.** Daniel Jurafsky and James Martin. 3rd edition. Chapter 8 'Sequence Labeling for Parts of Speech and Named Entities'[2], Chapter 9 'Deep Learning Architectures for Sequence Processing'[3]
3. **Large-Scale Language Models for Norwegian**[4]
4. **NorNE: Norwegian Named Entities**[5]
5. **'BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding'** Devlin et al 2019[6]

[1]`https://www.morganclaypool.com/doi/10.2200/S00762ED1V01Y201703HLT037`
[2]`https://web.stanford.edu/~jurafsky/slp3/8.pdf`
[3]`https://web.stanford.edu/~jurafsky/slp3/9.pdf`
[4]`http://norlm.nlpl.eu`
[5]`https://github.com/ltgoslo/norne`
[6]`https://www.aclweb.org/anthology/N19-1423/`

6. **‘NorNE: Annotating Named Entities for Norwegian’** Jørgensen et al 2020[7]
7. **‘Design Challenges and Misconceptions in Neural Sequence Labeling’** Yang et al 2018[8]

3 Introduction

This assignment deals with the task of named entity recognition (NER) and deep neural contextualized architectures likes ELMo and BERT. We *highly* recommend that you use Saga for development for this assignment, as resources on your own computers are unlikely to be sufficient for training or fine-tuning multiple models in time. You still can develop locally, if you want, but the resulting code is required to run OK on Saga with the default IN5550 modules loaded (*NLPL-nlptools/2021.01-gomkl-2019b-Python-3.7.4*, *NLPL-PyTorch/1.6.0-gomkl-2019b-Python-3.7.4*, and *NLPL-transformers/4.2.2-gomkl-2019b-Python-3.7.4*).

Please make sure you read through the entire assignment before you start. Solutions must be submitted via Devilry[9] by **23:59**, on the **12th of April, 2021**. Please upload a single PDF (4-8 pages) with details on your experiments and your answers to the questions in the assignment.

Your (heavily commented) code and data should be available in a separate private Git repository, on UiO’s hosted GitHub. Similar to the first assignments, make your repository private but allow IN5550 staff[10] to have access to it. The PDF in Devilry should contain a link to your repository.

If you have any questions, please raise an issue in the IN5550 Git repository[11], email *in5550-help@ifi.uio.no*, or ask on the Mattermost chat. Make sure to take advantage of the group sessions.

4 Named entity recognition for Norwegian

Named Entity Recognition (NER) is the task of identifying and categorising proper names in text. For this obligatory assignment, we will be using a dataset of named entity annotations for Norwegian (Bokmål): NorNE. The annotated corpus is published at `https://github.com/ltgoslo/norne/`, but **you have to use a modified version of i**t, with different train-dev-test splitting. It is available on Saga as `/cluster/projects/nn9851k/IN5550/norne-nb-in5550-train.conllu`. This file contains 18 098 annotated sentences, and you are free to divide into training and validation parts as you see fit. There is also a held-out test set of 1 940 sentences, which will be hidden until the assignment deadline.

[7]`https://www.aclweb.org/anthology/2020.lrec-1.559/`
[8]`https://www.aclweb.org/anthology/C18-1327/`
[9]`https://devilry.ifi.uio.no/`
[10]`https://github.uio.no/orgs/in5550/people`
[11]`https://github.uio.no/in5550/2021/issues`

NorNE adds named entity annotations to the Norwegian Dependency Treebank (NDT). The corpus contains a total of ∼300K tokens, of which ∼20K form part of a named entity annotation. The dataset annotates a rich set of entity types, including persons, products, organisations, location, and more – please consult the README file in the NorNE GitHub repository for more information about the corpus and the categories.

Conceptually, NER is a sequence segmentation task, and we want to identify sub-sequences with a given category, like in the following example (where `GPE_LOC` stands for *geo-political entity, location*):

	`ORG`					`GPE_LOC`	
Den	internasjonale	domstolen	har	sete	i	Haag	.
The	*International*	*Court of Justice*	*has*	*its seat*	*in*	*The Hague*	*.*

In practice, however, we approach the task as a sequence labelling problem, where we assign per-word labels according to some variant of the BIO ('beginning', 'inside', 'outside') scheme:

`B-ORG`	`I-ORG`	`I-ORG`	`O`	`O`	`O`	`B-GPE_LOC`	`O`
Den	internasjonale	domstolen	har	sete	i	Haag	.

NorNE is distributed in the tab-separated CoNLL-U format, where the final column specifies the named entity label using BIO format (in a variant sometimes referred to as BIO-2). The example from above then looks like this:

```
1 Den den DET _ Gender=Masc|Number=Sing|PronType=Dem 3 det _ name=B-ORG
2 internasjonale internasjonal ADJ _ Definite=Def|Degree=Pos|Number=Sing 3 amod _ name=I-ORG
3 domstolen domstol NOUN _ Definite=Def|Gender=Masc|Number=Sing 4 nsubj _ name=I-ORG
4 har ha VERB _ Mood=Ind|Tense=Pres|VerbForm=Fin 0 root _ name=O
5 sete sete NOUN _ Definite=Ind|Gender=Neut|Number=Sing 4 obj _ name=O
6 i i ADP _ _ 7 case _ name=O
7 Haag Haag PROPN _ _ 4 obl _ SpaceAfter=No|name=B-GPE_LOC
8 . $. PUNCT _ _ 4 punct _ name=O
```

The format is further explained in the README file of the NorNE GitHub repository. The most convenient tool for handling CoNLL-U files (both reading and writing) is the `CoNLL-U Parser` (`https://pypi.org/project/conllu/`). This Python package is available on Saga after loading the *NLPL-nlptools/2021.01-gomkl-2019b-Python-3.7.4* module. Remember that we provide the training data in CoNLL-U, and your prediction code should produce the predictions of your system in the same format, so that we could evaluate them on the held-out test set. Check that your CoNLL-U is valid before submitting your solution.

4.1 Evaluation

While NER could be evaluated by computing precision, recall and F1 at the *token-level*, it is more meaningful to evaluate at the *entity-level*. This can be done in several different ways, however, where the most strict (but also most common) approach would be to only consider a prediction to be correct if it is identical with the gold annotation, i.e. both the predicted boundary and entity type is correct.

This ('strict') is the approach that we employ in this assignment. Your submissions will be evaluated with the `eval_on_test.py` script[12]. It takes as an input two CoNLL-U files, one with your predictions on the test set, and another with the gold annotations. It then produces F1 scores for each named entity type and the overall F1 score across all types. Note that the tokens which are not part of any named entity (the `O` label) are omitted from the evaluation procedure. You can also look at this script to get an idea of how to work with CoNLL-U files in Python.

5 Sequence tagging with NorBERT

You are required to use a pretrained transformer-based language model for Norwegian: specifically, **NorBERT**. You should write this part of the obligatory using the `transformers`[13] toolkit; with it, you can trivially instantiate both the relevant tokenizer and the relevant model, both of which are available locally on Saga.[14] Load the model as follows:

```
norbert = "/cluster/shared/nlpl/data/vectors/latest/216/"
tokenizer = transformers.BertTokenizer.from_pretrained(norbert)
model = transformers.BertModel.from_pretrained(norbert)
```

The `transformers` Python package can be imported on Saga after loading the *NLPL-transformers/4.2.2-gomkl-2019b-Python-3.7.4* module.

You are to train a NER sequence tagger using contextual representations generated by NorBERT; to do so, you should use the default `BertModel` class, and build a tagger on top of it. You can also directly use the `BertForTokenClassification` subclass; this builds a minimal classifier on top of the pre-trained base model.

Fine-tuning

Experiment with fine-tuning. How different are your results when you allow NorBERT's parameters to be fine-tuned, compared to when you only train your classifier? How much longer does your system take to learn? Now, pick a subset of NorBERT's hyperparameters. This could, for example, be a few of the transformer layers, or a random selection of attention heads. Fine-tune only this subset. What did you expect would happen, re. both results and time? How is this different from what actually happens? Finally, you can try some more principled fine-tuning strategies, such as *chain-thaw* (Felbo et al.

[12]`https://github.uio.no/in5550/2021/tree/master/obligatories/3/eval_on_test.py`

[13]https://huggingface.co/transformers

[14]In principle, NorBERT is also available at `https://huggingface.co/ltgoslo/norbert`, but avoid downloading it again!

2017[15]) or *discriminative fine-tuning* (Howard and Ruder 2018[16]). Try out a few different combinations of parameters and describe your results.

Architecture

Experiment with different architectures built on top of NorBERT's last layer. The most basic architecture is of course just a linear layer, projecting NorBERT's representations into the NE label space. What happens if you add slightly more complex architectures? One option could be to plug NorBERT's representations into a recurrent network; another simpler one could be to just use multi-layer feed-forward-networks. Try out some of this architecture-building, and describe your results.

Motivate your choices, and describe the effect they had on your results. Report and discuss your results.

6 *Sequence tagging with NorELMo

This section is not mandatory; however, if you aren't confident about the previous section, this is an opportunity for you to get bonus points, up to a maximum of 6 points.

In this section, instead of using NorBERT, you are to use NorELMo, which is, as the name would imply, a Norwegian variant of ELMo. ELMo is a deep neural language model: like BERT, it produces contextualized token embeddings; unlike BERT it is trained using recurrent neural networks, not transformers.

The NLPL Vector repository currently provides 4 ELMo models trained on Norwegian corpora[17], of which we recommend NorELMo$_{30}$ and NorELMo$_{100}$. As a rule, ELMo models are used as feature extractors, without full fine-tuning. To actually extract contextualized ELMo embeddings for the word tokens in your data, we recommend to use the `simple_elmo` library[18]. On Saga, it is available after loading the *NLPL-simple_elmo/0.6.0-gomkl-2019b-Python-3.7.4* module.

7 Conclusion

Your final submission should include:

- a PDF report discussing your results,
- your best trained model for Norwegian NER,

[15]`https://www.aclweb.org/anthology/D17-1169/`
[16]`https://www.aclweb.org/anthology/P18-1031/`
[17]`http://wiki.nlpl.eu/Vectors/norlm/norelmo`
[18]`https://pypi.org/project/simple-elmo/`

- a `predict_on_test.py` script, taking as an input the path to the test CoNLL-U dataset and producing a version of the dataset with the existing named entity labels (if any) replaced with the predictions of your model:

  ```
  python3 predict_on_test.py --test <testfile>
  ```

- The resulting file must be named `predictions.conllu` and follow the same format as the test dataset (which in turn is exactly the same as the training set).

We will then evaluate your predictions against the gold labels with our own code (see subsection 4.1). Please make sure you include all necessary files in your submission. If you used any model not available locally on Saga, put it in the `/cluster/projects/nn9851k/IN5550/YOUR_USERNAME` directory. Do the same for your own trained sequence tagger if it is too heavy to be uploaded to the UiO GitHub.

If your code fails to run, you will lose points for the code section. You're supposed to provide sensible defaults, that load the appropriate model, and assume that they are located in the root directory. Feel free to give it a test run with a friend, before your final submission, and we will discuss this in a group session.

Good luck!