Integrating rules and neural nets for morphological tagging of Norwegian - Results and challenges

Anonymous Author
Affiliation / Address line 1
Affiliation / Address line 2
Affiliation / Address line 3
email@domain

Anonymouser Author
Affiliation / Address line 1
Affiliation / Address line 2
Affiliation / Address line 3
email@domain

Anonymousest Author
Affiliation / Address line 1
Affiliation / Address line 2
Affiliation / Address line 3
email@domain

Abstract

In this paper, we report on efforts to improve the Oslo-Bergen Tagger for Norwegian morphological tagging by using a hybrid system that combines the output of the rule-based Constraint Grammar tagger with a neural sequence-to-sequence model trained for tagging. The results are very promising for cases where the two systems intersect in tokenisation and morphological analysis, but problems remain in integrating the two systems in many cases.

1 Introduction

The Oslo-Bergen Tagger (OBT, Hagen and Johannessen 2003; Johannessen et al. 2012) is a widely used tool for morphological tagging of Norwegian text. It has existed in various incarnations for around 25 years, first as a purely rule-based system and later coupled with a statistical module for disambiguation. In this paper, we report on our recent efforts to bring the system into the age of neural networks and show that, even today, the rules boost accuracy considerably over a purely neural system, although there are challenges in combining rules and neural nets due to divergent tokenisations.

The structure of the paper is as follows: In section 2 we give some historical background on OBT and Section 3 describes the current status of its rule-based component. Section 4 describes the training and evaluation data that we have used in developing the new system. Section 5 then provides the details of how our neural system was trained while Section 6 describes how it was combined with the rule system. Section 7 evaluates the performance of the neural system alone as well as the combined system. Section 8 concludes.

2 History of the Oslo-Bergen Tagger

The Oslo-Bergen Tagger was originally developed between 1996 and 1998 by the Tagger Project at the University of Oslo. Rules for morphological and syntactic disambiguation were written in the first version of the Constraint Grammar framework (Karlsson et al., 1995), retrospectively called CG1. The rules were parsed by the only existing CG rule interpreter at the time, developed by Lingsoft AB. The input to CG disambiguation rules is multitagged text, i.e., text where each token has been annotated with all possible lexical analyses. Hence, the project also developed a lexicon with lemmas and inflected forms (later known as Norsk ordbank) and a combined tokenizer/multitagger. The tagger was developed for both Bokmål and Nynorsk, the two written varieties of Norwegian. In this article, we will only focus on the Bokmål version of the tagger, and only on the tokenizer and the morphological disambiguation.

The first version of the tagger was tested on an unseen evaluation corpus with a wide variety of text genres and achieved an F1-score of 97.2 (Hagen and Johannessen, 2003, 90). The numbers behind the F1-score – a precision of 95.4 and recall of 99.0 – reveal that the tagger leaves some ambiguity but makes relatively few errors. At the time, this was considered acceptable as the tagger was mostly used to annotate written corpora for linguistic research, where a high recall was considered more important than a high precision.

In 2000 the rule interpreter was replaced by a reimplementation in Allegro Common Lisp made by Paul Meurer in cooperation with the Text Laboratory at the University of Oslo. At the time, Meurer was employed at Aksis in Bergen, and hence the tagger was named the Oslo-Bergen Tagger (OBT).

Some years later the need for a new upgrade became urgent. Firstly, OBT was quite slow. This

was not a big problem in 2000, but soon our corpora were getting bigger, and speed became important. The project Norwegian Newspaper Corpus (2007–2009) gave the Text Laboratory the opportunity to translate the CG1 rules to the new more efficient and expressive CG3 format and to use a faster rule interpreter made by the VISL project at the University of Southern Denmark. Secondly, the ambiguities that were left in the output from OBT made the tagger unsuitable for many language technology purposes and applications that require the text to be completely disambiguated. We therefore extended OBT with a statistical module, implemented as a Hidden Markov Model, that disambiguated the remaining morphological ambiguities and also provided the system with a new feature: disambiguation of lemmas. The new OBT+Stat system achieved an accuracy of around 96 percent (Johannessen et al., 2012).

In the version of the tagger presented here, we have replaced the original HMM module with one that is based on neural networks. We do this for two reasons: First, the new module employs technology that has proven to yield superior results in a variety of NLP tasks. Secondly, the original module did not take into consideration the ambiguity left by the CG rules, meaning that the HMM might select a tag that was previously removed by the disambiguation rules or not even present in the tagger lexicon. The new machine learning module ranks possible readings by probability, allowing us to find the most probable reading (if any) in the intersection between its output and the remaining CG readings, hence not discarding the work that has already been done by the CG disambiguation rules if the intersection is non-empty, but leaving a question as to what to do if the intersection *is* empty.

3 The rule-based tokenizer and tagger

In this section, we first present some of the main tasks for the tokenizer and multitagger before we give a short description of the constraint grammar module. The tokenizer uses a lexicon with all possible lexical readings, where a reading is a combination of a lemma and a morphosyntactic tag chosen from a set of 149 possible analyses.[1] The lexicon was originally based on Norsk ordbank 2005,[2] but has since been updated with words more recently introduced into the language (such as *tvitre* 'tweet'). The newest version of the tokenizer is written in Python and mirrors in most cases the original tokenizer written in Perl. There is one major exception: The original system from the late '90s worked according to the strategy "Disambiguate as soon as possible" (Karlsson et al., 1995). This resulted in fixed expressions like *blant annet* ('among other things' - adverb) and *etter hvert* ('little by little' - preposition) being allowed – and disambiguated – in the lexicon. In the recent version of the tokenizer, such expressions are removed from the lexicon and the possible ambiguity is dealt with in the CG module. The main principle for the tokenizer is therefore to split tokens on blank space or a sentence delimiter like a full stop or a question mark. For each token identified, the original word form is rendered inside a `<word>`-tag and looked up in the lexicon. Non-sentence initial capitalized words are identified as proper nouns. Words that exist in the lexicon are assigned all readings found there. If the word is not found in the lexicon and not identified as a proper noun, the word is sent to a compound analyzer. Most unknown words will get an analysis here, as many of them are productively created compounds. Some words will still get the tag *ukjent* ('unknown') from the tokenizer. These words are often dialect words not standardized in the lexicon or foreign words. Figure A in the Appendix shows how the tokenizer and multitagger deals with the sentence *TV-programmet "Ut i naturen" begynner kl. 21.15.* ('The TV program "Ut i naturen" starts at 21.15.'), which has quotation marks, abbreviations, and a time expression.

The tokenizer also identifies sentences using sentence delimiters. A list of known abbreviations and linguistic rules, like the rule "the word including the full stop character is an abbreviation if the word is in the abbreviation list or if the following word is not capitalized", identifies abbreviations like *kl.* (abbreviation for "o'clock" used to specify time in Norwegian) in Figure A. Headlines are also identified by rules and get their own tag.

The constraint grammar module takes tokenized and multitagged text as input and its main task is to reduce the number of readings to ideally one per word. The number of readings left by the multitag-

[1] The complete list is available at `http://tekstlab.uio.no/obt-ny/morfosyn.html`.

[2] `https://www.nb.no/sprakbanken/en/resource-catalogue/oai-nb-no-sbr-5/`

ger varies a lot. In the test corpus used in this article (which will be further described in Section 4) there are on average 2,04 readings per word. After the CG rules are applied, there are on average 1,09 readings left per word.

Figure B in the Appendix shows the output from the CG module in debug mode for the sentence *Rosa cupcakes hører kanskje med når man skal ha bloggtreff?* ('Pink cupcakes might be part of a blog meeting?'). Readings that have been removed starting with ";" and the ID numbers of the rules applied are appended to each reading. Note that the English loan word *cupcakes* is not identified in the lexicon or in the compound analyzer and has got the tag *ukjent* 'unknown'. The compound *bloggtreff* 'blog meeting' was not in the lexicon but has got two readings from the compound analyzer. As the examples show, there are both REMOVE rules (remove a reading) and SELECT rules (select a reading). A rule can be very simple, like rule 2430 in Figure 1 that says "select the verb infinitive reading if the verb to the left is a modal auxiliary and not in the set of dangerous infinitives (= not likely infinitives)".

```
#:2430
SELECT:2430 (verb inf) IF
(NOT 0 farlige-inf)
(-1 m-hj-verb)
;
```

Figure 1: Simple SELECT rule

Figure 2 shows an example of a more complex rule with linked context conditions somewhere to the right in the sentence. The rule says: "choose the subjunction reading - if somewhere to the right there is a safe noun or pronoun (stop looking if a word on the way has a reading that is not an adverb, adjective or determinative) – and – if there is a word in the present or past tense after the noun/pronoun (adverbs between are fine)."

```
#:2579
SELECT:2579 (sbu) IF
(...)
(**1C subst/pron BARRIER
   ikke-adv-adj-det)
(**1C subst/pron LINK *1
   ikke-adv LINK 0 pres/pret)
;
```

Figure 2: More complex SELECT rule

The CG grammar for Bokmål has more than 2300 rules. 1995 of them are SELECT rules. Some rules apply to all possible words, while some are rules for specific word forms. When the original CG grammar was developed, a training corpus of 100 000 words from novels, newspapers and magazines was used. For each new rule added to the grammar, we checked how the rule worked by looking at recall and precision. Most rules remove or choose readings without making too many errors. But in the last period of the project, we made around 250 heuristic rules to speed up the disambiguation. These rules were riskier but in our small training corpus, they worked well. Later in this article, we will see whether the combination of the CG rules and the neural net is affected if the heuristic rules are removed from the grammar.

4 Training and evaluation data

The training and evaluation corpus that was used in earlier stages of development of the OBT system is no longer suitable because the tagset and the tokenisation principles have evolved. Instead of bringing this corpus up to date, we chose to use the Norwegian Dependency Treebank (NDT, Solberg et al. 2014) in the development of the new version of OBT. The Bokmål part of NDT is around 300 000 tokens and consists of blog text, news text, parliament proceedings and government white papers.

The NDT CoNLL data were converted to the format of the OBT. We also extracted the pure text and ran OBT on it without statistical disambiguation, to compare the outputs. If the NDT analysis was not among the analyses produced by OBT, we either corrected the NDT annotation if that was the source of the error, or changed the rules of the OBT system if that could easily be done. This process was iterated a few times. Notice that during this period, the whole data set was used for development, as is common with rule-based systems. The goal was to improve both the accuracy of the rule-based disambiguation and the quality of the training data for the neural component.

The performance of the rule-based system by the end of this phase is shown in Table 1. When heuristic rules are used, we see that in 7.5% of cases, OBT produces an ambiguous analysis containing the correct tag as one possibility, whereas 1.8% of tokens are only given (one or more) wrong analyses. Disabling the heuristic rules reduces the number of wrong tags by 0.2% but at the cost of an increase of 3.3% of tokens that get an ambiguous

analysis containing the correct tag.

The role of the statistical system is to pick the correct analysis in the ambiguous cases. On its own the neural net might be able to predict the right analysis even in cases where the rules are wrong. However, this analysis will be discarded when we intersect its output with the rules.

with heuristic rules		
unambiguous correct	280650	(90.7%)
ambiguous incl. correct	23219	(7.5%)
wrong	5413	(1.8%)
without heuristic rules		
unambiguous correct	270830	(87.6%)
ambiguous incl. correct	33597	(10.8%)
wrong	4855	(1.6%)

Table 1: Performance of the rule-based system

For the training of the neural system, we then split the corpus into train-dev-test sets. While doing this, we made sure the output tags in the training set covered all output tags in the dev and test sets to ensure that the model was trained with samples from all tags. We do this by, first, initializing the Python random seed as 0, then, splitting the data and checking if the training set covers all tags. If it does not, we increase the random seed by one and do the same until we find a training set that covers all the tags in the other sets. This way, we randomly split the dataset into 80-10-10 percent partitions to obtain train-dev-test datasets respectively.

Finally, the data was reformatted for the neural network. Figure 3 shows an example of input and output for a sentence. The input is the tokenized form of the sentence. The output is the sequence of serialized tags for each token in the input. The token <next_token> is an indicator that all tags of the corresponding input token have finished and tags of the next input token start afterward.

```
INPUT: Men det er bare noe jeg tror .

OUTPUT:
:konj: clb <next_token>
:pron: 3 ent nøyt pers <next_token>
:verb: pres <next_token>
:adv: <next_token>
:pron: 3 ent nøyt pers <next_token>
:pron: 1 ent hum nom pers <next_token>
:verb: pres <next_token>
$punc$ :clb: <punkt>
```

Figure 3: An example input and output for a sentence.

5 The neural system

Recently, a BERT (Devlin et al., 2018) pre-trained encoder (nb-bert-base) was published by the Norwegian National Digital Library (Kummervold et al., 2021). This pre-trained encoder for Norwegian provides a rich feature set that was previously lacking for the language. Furthermore, since the tagged corpus is very small in comparison to the corpus the pre-trained model was trained on, it is important to use the pre-trained model in order to be able to generalize to unseen data. Therefore, we follow an approach similar to that of Omelianchuk et al. (2020) and use a sequence-to-sequence (seq2seq) setting to tag the sentences using the pre-trained model.

Sequence-to-sequence models have two main components: an encoder and a decoder. The encoder side is set as the encoder nb-bert-base (NbAiLab, 2021). For the decoder, we randomly initialize 6 layers of size 768 with 12 attention heads. The decoder also has cross-attention layers as it was shown to be effective in seq2seq training (Gheini et al., 2021). We freeze the encoder weights throughout the training since using the encoder as a feature extraction mechanism in this way was shown to be beneficial (Zoph et al., 2016) and is a common practice (Gheini et al., 2021). We use the EncoderDecoderModel provided by the HuggingFace transformers library (Wolf et al., 2020) to configure and train a model.

The encoder-decoder model gets its input as the identifiers of the tokens (token numbers) in the input vocabulary and outputs the token numbers in the output vocabulary. Thus, the input and output are tokenized using these vocabularies. Since the encoder model had already been trained (nb-bert-base) using the widely-utilized sub-word tokenizer Wordpiece (Wu et al., 2016), we use that tokenizer as provided by the Huggingface Tokenizers library. For the decoder side, since our vocabulary size is very small and obvious (82 tags and 5 extra special tokens such as [CLS] and [SEP]), we do not need to train a special tokenizer. We define the vocabulary manually with these output tokens for use by the Wordpiece tokenizer.

The training configuration is as follows: We use the Adam optimizer (Kingma and Ba, 2015) with a learning rate of 0.0001. We set the batch size to 16 sentences as this is the amount the graphic cards could handle. We use the negative log-

likelihood loss (Yao et al., 2020) to compute the loss in each batch between the model output and the expected output. For any other parameter not mentioned in this section, we use the default value defined by version 4.17.0 of the Transformers library in the objects of the following types: BertConfig, EncoderDecoderModel, EncoderDecoderConfig, and BertModel.

We evaluate the model using the dev set during the training. We do this by using the BLEU score (Papineni et al., 2002) that is widely utilized to evaluate seq2seq models. We compute the BLEU score between the expected output and the model output for each sentence. We get the average of these scores for the whole dev set. We run the training for 300 epochs and keep the model that results in the maximum average BLEU score for the dev set.

6 Combining neural nets and rules

As mentioned in section 2, the current system prefers tags that are found in the intersection between the output of the CG rules and that of the neural network. Ideally, we would be able to find such intersections for each individual token separately. However, since the probability of a reading for a particular token depends on the selected readings for all other tokens in the sentence, the only viable option is to consider readings for entire sentences. Thus, for each input sentence, we find the list of possible readings produced by the network and calculate its probability. Then for each reading in this list, ordered by decreasing probability, we go through each token and check whether the tag assigned by the network is also found among those left by the CG disambiguation rules. If it is not found, we skip to the next reading in the list. If it is found, we go on to check the next token, and so on until we reach the end of the sentence, at which point the reading is picked as the selected one for the sentence. For the present test set, we find intersecting tags for all tokens for 1412 of the 2003 sentences (70.5%). The cases with missing intersections may be due to differences in either tokenisation (205 cases) or tag assignments (386 cases) between the two systems. When the tokenisations are different, it is not clear what to do. But if the tokens are the same, but the tag assignments differ, we can default to the most probable reading in the neural net output. We explore this option in Section 7.2.

Figure 4 shows a case where the tokenisation of the neural system does not match with the gold data in the test set. The neural system has split the initial, unknown proper name at a hyphen, whereas the CG tagger keeps it as one token. Since tokenisation is part of a preprocessing step and misalignments in tokenisation is a problem to be solved separately from tag assignment, in this paper we focus primarily on cases where the two systems do produce matching tokenisation, while improving the tokenisation match will be part of future work.

```
Neural net: Garosu - gil , som betyr [...]
CG: Garosu-gil , som betyr [...]
```

Figure 4: Mismatching tokenisation

```
Neural net:
Rosa        adj fl pos
cupcakes    subst appell fl mask ub  <---
hører       verb pres
kanskje     adv
med         prep
når         sbu
man         pron ent hum nom pers
skal        verb pres
ha          verb inf
bloggtreff  subst appell ent nøyt ub
?           clb <spm>
CG:
Rosa        adj fl pos
            adj nøyt ub ent pos
            adj ub m/f ent pos
            subst appell ubøy
            subst appell fem be ent
cupcakes    ukjent                   <---
hører       verb pres
kanskje     adv
med         prep
når         sbu
man         pron ent hum nom pers
skal        verb pres
ha          verb inf
bloggtreff  subst appell ent nøyt ub
            subst appell fl nøyt ub
?           clb <spm>
```

Figure 5: Non-intersecting tags

Figure 5 shows the problem of mismatching tags. For the first word, the CG tagger has left five possible analyses, and the neural net has correctly disambiguated to the plural adjective reading. However, OBT did not recognize the second word, *cupcakes*, and has therefore left an *ukjent* ('unknown') tag while the neural system has no analysis with that tag. Instead, the most probable analysis of the sentence according to the neural net has *cupcakes* correctly as an indefinite plural

system	accuracy
pure ML	96.9%
OBT + ML	99.0%
OBT w/o heur. + ML	99.0%

Table 2: Accuracy of different systems, sentences with intersecting tags

noun. However, since tag probabilities are conditional on all other tags in the sentence these two analyses are incomparable: it is not safe to disambiguate the CG analysis of *rosa* based on this analysis from the neural net, especially not when the mismatching tag is on the neighbouring word *cupcakes*.

In this particular case, the neural net is correct in its analysis of *cupcakes*. In general, it might be safe to assume that the neural system is correct in cases where the CG tagger assigns *ukjent*, and this is an option we will pursue in future research. However, as we will see in the Section 7 the neural system is often incorrect in cases where the tags do not intersect. Solving this problem may require more training data or fine-tuning the parameters of the tag generation process of the decoder of the seq2seq model.

7 Evaluation and error analysis

7.1 Sentences with intersecting tags

We first focus on the restricted cases where the ML system and the CG grammars not only have matching tokenisations but also intersecting tags. We evaluate three different setups: 1. the trained neural net used as a stand-alone morphological tagger 2. the rule-based system intersected with the neural net as described in Section 6 3. as the previous, but without the heuristic rules.

The performance of the three systems is shown in Table 2. Because we evaluate on intersecting tags only, the numbers do not show the actual performance of the system on running text. They do however clearly show that in the 70.5% of cases where the tags intersect, the rules strongly improve the performance of the systems: two-thirds of the tokens that are mistagged by the neural net now get a correct analysis. We also see that it makes no difference whether we run the system with or without the heuristic rules: the reduction of wrong tags that we saw in Table 1 is balanced out by the increase in ambiguity. On the sentences where this setup works, the performance is extremely good at an accuracy of 99.0%. By contrast, the widely used Spacy tagger reports an accuracy of 95.0% for morphological tagging of Norwegian UD.[3]

Since removing the heuristic rules gave no increase in performance, we focus on the setup with the full rule set in the following. This system mistags 184 tokens (out of 18612 in total in the matching sentences of the test set), whereas the pure ML system mistags 565 tokens. However, the error profile of the two systems is quite different, suggesting possibilities for further improvement.

Tables 3 and 4 show the twelve most common error types of the systems. We see that a relatively common error in the OBT + ML system involves perfect participles which often coexist with homonymous adjectives in Norwegian (as in other Germanic languages, cf. English 'bored') with often very slight or no semantic difference. OBT+ML overapplies the adjective analysis (in three different varieties) compared to the gold data, for a total of 14+10+6=30 errors. By contrast, the ML system on its own makes only 8+8=16 errors of this kind, suggesting that the rules disambiguate wrongly. Performance might therefore increase if we leave this decision to the neural net, though it is worth mentioning that this system makes 6 errors in the opposite direction (which only happens twice when the rules are used and therefore does not show up in the table). Apart from errors with participles, all other frequent errors involve gender assignment or number assignment on indefinite neuter nouns. The latter distinction is hard to make because these indefinite neuters make no morphological distinction between singular and plural and the context is not always clear. As for the gender errors, at least some of these are errors in the gold tags that were not caught in our manual correction. The feminine/masculine distinction has disappeared in the Oslo dialect of Norwegian (Lødrup, 2013) and it may have been hard for the annotators to choose the correct tag. Another debatable case is gender assignment on proper nouns, which is often missing from the ML system output, but is also not systematic in the gold data. Here it may be better to just standardise on not assigning gender to proper nouns.

[3] See `https://spacy.io/models/nb`. As the Norwegian UD corpus (Øvrelid and Hohle, 2016) is an automatic conversion of the NDT corpus, the complexity of the tasks should be comparable, although the test split is not identical.

Gold tag	Predicted tag	Freq
[':verb:', 'perf-part']	[':adj:', '<perf-part>', 'ent', 'm/f', 'ub']	14
[':subst:', 'appell', 'ent', 'mask', 'ub']	[':subst:', 'appell', 'ent', 'fem', 'ub']	13
[':verb:', 'perf-part']	[':adj:', '<perf-part>', 'ent', 'nøyt', 'ub']	10
[':adj:', 'ent', 'nøyt', 'pos', 'ub']	[':adj:', 'ent', 'm/f', 'pos', 'ub']	10
[':subst:', 'appell', 'fl', 'mask', 'ub']	[':subst:', 'appell', 'fem', 'fl', 'ub']	9
[':subst:', 'appell', 'ent', 'nøyt', 'ub']	[':subst:', 'appell', 'fl', 'nøyt', 'ub']	8
[':verb:', 'perf-part']	[':adj:', 'ent', 'm/f', 'pos', 'ub']	6
[':subst:', 'appell', 'be', 'fl', 'mask']	[':subst:', 'appell', 'be', 'fem', 'fl']	5
[':subst:', 'appell', 'be', 'ent', 'mask']	[':subst:', 'prop']	5
[':pron:', '3', 'fl', 'pers']	[':det:', 'fl', 'kvant']	4
[':subst:', 'appell', 'ent', 'mask', 'ub']	[':subst:', 'appell', 'ent', 'nøyt', 'ub']	4
[':subst:', 'appell', 'fl', 'nøyt', 'ub']	[':subst:', 'appell', 'ent', 'nøyt', 'ub']	4

Table 3: Most frequent errors, OBT + ML

Gold tag	Predicted tag	Freq
[':subst:', 'appell', 'ent', 'mask', 'ub']	[':subst:', 'appell', 'ent', 'fem', 'ub']	13
[':adj:', 'ent', 'nøyt', 'pos', 'ub']	[':adj:', 'ent', 'm/f', 'pos', 'ub']	12
[':subst:', 'appell', 'fl', 'mask', 'ub']	[':subst:', 'appell', 'fem', 'fl', 'ub']	10
[':verb:', 'perf-part']	[':adj:', '<perf-part>', 'ent', 'm/f', 'ub']	8
[':verb:', 'perf-part']	[':adj:', '<perf-part>', 'ent', 'nøyt', 'ub']	8
[':subst:', 'mask', 'prop']	[':subst:', 'prop']	8
[':subst:', 'appell', 'ent', 'nøyt', 'ub']	[':subst:', 'appell', 'fl', 'nøyt', 'ub']	8
[':subst:', 'appell', 'ent', 'mask', 'ub']	[':subst:', 'appell', 'ent', 'nøyt', 'ub']	8
[':subst:', 'appell', 'ent', 'fem', 'ub']	[':subst:', 'appell', 'ent', 'mask', 'ub']	7
[':subst:', 'appell', 'be', 'fl', 'mask']	[':subst:', 'appell', 'be', 'fem', 'fl']	6
[':subst:', 'appell', 'ent', 'mask', 'ub']	[':prep:']	6
[':adj:', '<perf-part>', 'ent', 'm/f', 'ub']	[':verb:', 'perf-part']	6

Table 4: Most frequent errors, ML system (intersecting tags only)

Gold tag	Predicted tag	Freq
[':adj:', 'ent', 'nøyt', 'pos', 'ub']	[':adj:', 'ent', 'm/f', 'pos', 'ub']	24
[':subst:', 'appell', 'ent', 'mask', 'ub']	[':prep:']	24
[':prep:']	[':subst:', 'appell', 'ent', 'mask', 'ub']	19
[':verb:', 'pres']	[':prep:']	18
[':subst:', 'appell', 'ent', 'mask', 'ub']	[':subst:', 'appell', 'ent', 'fem', 'ub']	18
[':prep:']	[':subst:', 'prop']	17
[':subst:', 'appell', 'ent', 'fem', 'ub']	[':subst:', 'appell', 'ent', 'mask', 'ub']	16
[':prep:']	['$punc$', ':<komma >:']	15
[':prep:']	[':verb:', 'pres']	14
[':subst:', 'appell', 'fl', 'mask', 'ub']	[':subst:', 'appell', 'fem', 'fl', 'ub']	14
[':subst:', 'mask', 'prop']	[':subst:', 'prop']	14
[':subst:', 'appell', 'ent', 'mask', 'ub']	[':subst:', 'appell', 'ent', 'nøyt', 'ub']	14

Table 5: Most frequent errors, ML system (all matching tokenisations)

system	accuracy
pure ML	92.8%
OBT w/o heur. + ML	94.1%

Table 6: Accuracy of different systems, all sentences with matching tokenisation

7.2 All sentences with matching tokenisations

To test whether the neural system can be trusted in cases where there is no overlap in tag assignment, we also evaluate the system on all sentences where the tokenisation is matching. We test two setups: one where we use the (non-heuristic) rules plus the neural system as described above, but default to the output of the neural tagger in cases where there is no overlap, and one where we only use the best ML tag. The performance of the two setups is given in Table 6

As we can see, the results drop considerably. Overall performance is now below that of the Spacy tagger. Put another way: when we evaluate all sentences with matching tokenisations, the size increases by 8036 tokens from 18612 to 26648, but the number of errors increases from 565 to 1940, i.e. 1375, indicating an error rate of 17.1% on the tokens where the intersection with the output of the CG tagger is empty. Table 5 shows the frequency of errors, which looks very different from Table 4. Most strikingly, there are now many errors involving the part-of-speech tag :prep: (preposition), which is both over- and underpredicted by the system. Prepositions are a closed class in Norwegian, as in many other languages, and so it is surprising that the system goes wrong in so many cases here.

We used an encoder-decoder model to generate the tags given a sentence. This is a different approach from the majority of the work on tagging using deep learning, where the task is formalized as a sequence classification task. We have chosen to use this architecture as we have 82 tags in the gold data that would require training many sequence classifiers or a single classifier that would require many classes (tag combinations)[4] to be trained on. Since there are many layers between the input and output of our model (12 Bert, and 6 decoder layers), the model sometimes misses the syntactic alignment between the input and the output. This is, we believe, the main reason for the mismatches.

For future work, we focus on solving the issues with mismatching and incorrect tagging. We plan to use accuracy as the evaluation metric to select the best-performing model using the dev set. In addition, we plan to use various constraining configurations of beam search on generating tags. In our experiments, we observed that beam search considerably slowed down the evaluation on the dev set resulting in an overall performance drop in the training process. Thus, we plan to experiment with the performance of a beam search-based evaluation by applying it for various epoch intervals but not all intervals. And finally, we plan to pick the best tag-set from the output of beam-search by introducing manual rules to avoid mismatching.

8 Conclusion

We have presented a hybrid system for tagging Norwegian texts, based on intersecting the output of a rule-based Constraint Grammar system and a neural sequence-to-sequence model based on a large, pre-trained language model. Our results so far indicate that there are both great opportunities and considerable challenges in making such a system work.

On the plus side, we observe that when the tokenisations of the two systems match and the intersection of the possible analyses is non-empty, performance is extremely good at 99.0%. On the downside, it is challenging to make the two systems work together; in about 10% of cases, the tokenisation does not match, and in around 20% of cases, the intersection of analyses is empty. We have seen that in some cases, it is tempting to let the neural system overrule the rules, but overall its performance in these cases is not good. Hence our overall priority in future work will be to improve the neural system.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. `http://arxiv.org/abs/1810.04805`.

Mozhdeh Gheini, Xiang Ren, and Jonathan May. 2021. Cross-attention is all you need: Adapting pretrained Transformers for machine translation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1754–1765, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

[4]See the tag combinations: http://tekstlab.uio.no/obt-ny/english/morphosyn.html

Kristin Hagen and Janne Bondi Johannessen. 2003. Parsing nordic languages (panola) – norsk versjon. In Henrik Holmboe, editor, *Nordisk Sprogteknologi 2002*, pages 89–96. Museum Tusculanum, Copenhagen.

Janne Bondi Johannessen, Kristin Hagen, André Lynum, and Anders Nøklestad. 2012. Obt+stat. In Gisle Andersen, editor, *Exploring Newspaper Language: Using the web to create and investigate a large corpus of modern Norwegian*, pages 51–66. John Benjamins, Amsterdam.

Fred Karlsson, Atro Voutilainen, Juha Heikkilä, and Arto Anttila, editors. 1995. *Constraint Grammar: A Language-Independent Framework for Parsing Unrestricted Text*. Mouton de Gruyter, Berlin.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Per E Kummervold, Javier De la Rosa, Freddy Wetjen, and Svein Arne Brygfjeld. 2021. Operationalizing a national digital library: The case for a Norwegian transformer model. In *Proceedings of the 23rd Nordic Conference on Computational Linguistics (NoDaLiDa)*, pages 20–29, Reykjavik, Iceland (Online). Linköping University Electronic Press, Sweden.

Helge Lødrup. 2013. Hvor mange genus er det i oslodialekten? *Maal og Minne*, 103(2).

NbAiLab. 2021. Norwegian Transformer Model. `https://github.com/NbAiLab/notram/tree/0c90d6b28008df514c4ac847e4c9d68f4709a181`, Accessed: 12.12.2022.

Kostiantyn Omelianchuk, Vitaliy Atrasevych, Artem Chernodub, and Oleksandr Skurzhanskyi. 2020. GECToR – grammatical error correction: Tag, not rewrite. In *Proceedings of the 15th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 163—-170. Association for Computational Linguistics.

Lilja Øvrelid and Petter Hohle. 2016. Universal dependencies for norwegian. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1579–1585.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Per Erik Solberg, Arne Skjærholt, Lilja Øvrelid, Kristin Hagen, and Janne Bondi Johannessen. 2014. The Norwegian dependency treebank. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 789–795, Reykjavik, Iceland. European Language Resources Association (ELRA).

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. 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*.

Hengshuai Yao, Dong-lai Zhu, Bei Jiang, and Peng Yu. 2020. Negative log likelihood ratio loss for deep neural network classification. In *Proceedings of the Future Technologies Conference (FTC) 2019*, pages 276–282, Cham. Springer International Publishing.

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer learning for low-resource neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1568–1575, Austin, Texas. Association for Computational Linguistics.

Appendix: sample multitagger and CG output

```
<word>Tv-programmet</word>
"<tv-programmet>"
"tv-program" subst appell nøyt be ent
 samset-leks <*program> <+programmet>
<word>«</word>
"<«>"
"$«" <anf>
<word>Ut</word>
"<ut>"
"ut" prep
"ut" adv
<word>i</word>
"<i>"
"i" prep
"i" subst appell mask ub ent
<word>naturen</word>
"<naturen>"
"natur" subst appell mask be ent
<word>»</word>
"<»>"
"$»" <anf>
<word>begynner</word>
"<begynner>"
"begynne" verb pres
"begynner" subst appell mask ub ent
<word>kl.</word>
"<kl.>"
"kl." subst appell fork
<word>21.15</word>
"<21.15>"
"21.15" subst <klokke>
"21.15" det kvant
<word>.</word>
"<.>"
"$." clb <<< <punkt> <<<
```

Figure A: Tokenized and multitagged sentence

```
<word>Rosa</word>
"<rosa>"
"rosa" adj fl pos
"rosa" adj nøyt ub ent pos
"rosa" adj ub m/f ent pos
"rosa" subst appell ubøy
"rose" subst appell fem be ent
; "rosa" adj be ent pos REMOVE:2311
<word>cupcakes</word>
"<cupcakes>"
"cupcakes" ukjent
<word>hører</word>
"<hører>"
"høre" verb pres
<word>kanskje</word>
"<kanskje>"
"kanskje" adv
<word>med</word>
"<med>"
"med" prep
<word>når</word>
"<når>"
"når" sbu SELECT:2579
; "nå" verb pres SELECT:2579
; "når" adv REMOVE:3383
<word>man</word>
"<man>"
"man" pron ent pers hum nom
 SELECT:3451
; "man" subst appell fem ub ent
 SELECT:3451
; "man" subst appell mask ub ent
 SELECT:3451
; "mane" verb imp SELECT:3451
<word>skal</word>
"<skal>"
"skulle" verb pres <aux1/perf_part>
 <aux1/infinitiv>
<word>ha</word>
"<ha>"
"ha" verb inf <aux1/perf_part>
 SELECT:2430
; "ha" interj SELECT:2430
; "ha" subst symb REMOVE:3574
; "ha" verb imp <aux1/perf_part>
SELECT:2430
<word>bloggtreff</word>
"<bloggtreff>"
"bloggtreff" subst appell nøyt ub ent
 samset-analyse <+treff>
"bloggtreff" subst appell nøyt ub fl
 samset-analyse <+treff>
<word>?</word>
"<?>"
"$?" clb <<< <spm> <<<
```

Figure B: Tokenized, multitagged and disambiguated sentence