Targeted Sentiment Analysis on $NoReC_{fine}$ dataset using pre-trained language models and RNNs

Rohullah Akbari
University of Oslo
rohullaa@uio.no

Liang Jia
University of Oslo
liangj@uio.no

Ece Cetinoglu
University of Oslo
ecec@uio.no

Abstract

In this paper, we applied different neural network architectures in order to classify the targeted sentiment in the $NoReC_{fine}$ dataset[1]. The architectures we have used are recurrent neural network such as Gated recurrent unit (GRU) and Long short-term memory (LSTM), the Norwegian ELMo ($NorELMo_{30}$), pre-trained transformer-based language models such as the NorBERT and the Multilingual BERT. We trained these models with and without including character-level information and encoding types of BIO and BIOUL. The pre-trained models significantly improve the performance compared with RNNs. The best model was found to be the NorBERT on the BIO encoded dataset with F_1 at **0.644** (binary) and **0.513** (proportional).

1 Introduction

Sentiment Analysis (SA) is a research field within Natural Language Processing (NLP) that aims to identify and categorize opinions in text and determine whether they are written in a positive, negative, or neutral tone. However, there could be more than one sentiment towards different targets in one sentence. A fine-grained sentiment analysis or targeted sentiment analysis (TSA) is used to solve this problem, which is identifying opinions in sentences and analyzing them regarding their polar expressions, targets, and holders. For example, the sentence below has only one sentiment towards one target, where the *normal* sentiment analysis model works fine.

(1) Denne disken$_{\text{POS}}$ er svært stillegående
'*This disk is very quiet-going*'

But the second sentence has two different sentiments towards two different targets, where normal sentiment analysis cannot identify its sentiments.

(2) En klassisk, men like så episk oppbygget inspo-pop som fengsler i refrenget$_{\text{POS}}$, men Perry$_{\text{NEG}}$ sliter med den dypere vokalen i versene.
'*A classic, but just as epic built-up inspo-pop that captivates with the chorus but Perry struggles with the deeper vocals in the verses.*'

The aim of this paper is to develop machine learning models that are able to classify these types of sentiments. In order to do so, we will apply a range of different neural network architectures, such as

(1) Recurrent neural network
(2) ELMo
(3) BERT models

The selected models for Recurrent neural networks (RNN) and pretrained models were also tested with included character-level information. The two best models are then applied with another more fine-grained label encoding of BIOUL.

2 The dataset

In this task we are working with the $NoReC_{fine}$ dataset (Lilja Øvrelid and Velldal, 2020) which is a dataset for fine-grained sentiment analysis in Norwegian. We have three separate data files for training, validating, and testing. The size of the dataset was shown in the table [1]. It has been annotated with respect to polar expressions, targets, and holders of opinion. The distribution of different polar labels was shown in table [2]. The dataset is encoded in **B**egin **I**nside **O**utside (BIO) type of encoding. In total, there are five classes in the data

(1) **O**: outside
(2) **B-targ-Positive**: beginning of a target with

[1]The code for this paper can be found at https://github.uio.no/ecec/TSA. Read the README.md for how to run the code.

positive sentiment
(3) **I-targ-Positive**: inside of a target with positive sentiment
(4) **B-targ-Negative**: beginning of a target with negative sentiment
(5) **I-targ-Negative**: inside of a target with negative sentiment

An example of a sample from the dataset is shown in table [3].

Train	Test	Development	Total
8633	1271	1530	11434

Table 1: Dataset size of training, test and development sets

Label	Train	Test	Dev	Total
B-neg	1558	210	256	2024
B-pos	3486	525	620	4631
I-neg	1472	163	244	1879
I-pos	3399	572	581	4552
O	134317	20355	24195	178867
Neg (total)	3030	373	500	3903
Pos (total)	6885	1097	1201	9184

Table 2: Number of each tag appearing in training, test and development sets.

We observe that the dataset mostly consists of words with neutral (O-tagged) sentiment, with very few words with positive sentiment and even fewer words with negative sentiment (table [2]). It seems that the separation of training, development, and test datasets has already considered the imbalance of different labels and they have a similar distribution for different labels.

Han	B-targ-Neg
redda	O
den	O
perfekte	O
hustrue-skuespilleren	O
Bonnie	B-targ-Pos
Bedelia	I-targ-Pos
og	O
blødde	O
overfladisk	O
.	O

Table 3: Sample 99 from training dataset. Here, the *B-targ-Neg*, *B-targ-Pos* indicate the beginning of a token which has been annotated negative and positive, *I-targ-Pos* indicate the inside of a token with positive annotation and the *O* indicate the outside (neutral) token.

3 Architectures and Methodologies

A diverse collection of models were trained and tested for the task. All models were fine-tuned at the end. When the hyperparameters which yield the best results were found for each model, we tested the models with the optimal hyperparameters for different encodings of the gold labels and/or additional embeddings which provide character-level information.

3.1 Recurrent neural networks (RNNs)

Recurrent Neural Networks (RNNs) like Gated recurrent unit (GRU) and Long-short term memory (LSTM) have been proved to be effective in TSA and bringing significant improvements (Zhang et al., 2016; Yukun Ma and Cambria, 2018, 2019). Therefore, we are interested in how these networks deal with the $NoReC_{fine}$ dataset. For all of the RNNs, we used the static embedding with an ID of 58 trained on the Norwegian-Bokmaal CoNLL17 corpus from NLPL [2].

3.1.1 Baseline

The baseline model for this task was a simple bidirectional long short-term memory (BiLSTM) neural network with a linear output layer. We have later on adjusted this model such that it could be used with different versions of encodings for the gold labels, or with embeddings that provide character-level information.

[2]The embedding is available at http://vectors.nlpl.eu/repository/20/58.zip

3.1.2 GRUs and LSTMs

We have built two different recurrent neural network models for this task, and tested them for a variety of hyperparameters. Firstly, we improved the baseline model and ensured that the model had some additional properties and parameters to be tuned. The baseline model was a simple bidirectional LSTM, whereas our improved model can be bidirectional or non-bidirectional, can use BIOUL encoded target labels, can add character-level information on top of the pre-trained embeddings, and most common properties of a neural network, such as the size of the hidden dimensions or the number of layers, can be tuned. Additionally, a GRU model can be trained with the same properties.

3.2 Embeddings from Language Model (ELMo)

The problem with static embeddings, section 3.1, is that these embeddings assign the same vectors to same words in different contexts. This leads to incorrect classification. The solution to this is to use the Embeddings from Language Model (ELMo) which assigns each word a vector which is a function of the whole input phrase (Oren Melamud, 2016; Bryan McCann, 2017). In this way, the words are context-dependent, meaning that ELMo produces different representations for words that share the same spelling but have different meanings. ELMo is based on bi-directional LSTMs to process character-level tokens and generate word-level embeddings (ELMo, 2022).

In this project, we implemented the NorELMo (ID 217) which is an ELMo model for Norwegian[3]. This model is a set of bidirectional recurrent ELMo language models trained on Norwegian Wikipedia texts. The implementation is done by using the *allennlp* package [4]. The complete architecture for this section consists of NorELMo as the first layer, where it extracts word embeddings from the given text-data, and on top of this layer we have implemented a GRU model.

3.3 Transformer models

Bidirectional Encoder Representations from Transformers (BERT) is a transformer-based model that relies purely on the self-attention mechanism, which is made possible by a deep bidirectional Transformers at its core. This is essential since a word's meaning can often alter as a phrase progresses. BERT is able to process words in relation to all the other words in a sentence by looking at a word and all the other words that come before and after it at the same time. Because of this, the BERT models have become very popular in the NLP world in recent years, due to their good performance on most NLP tasks.

Another attempt has been made to run BERT models in the $NoReC_{fine}$ dataset. One example is from a master's thesis (Rønningstad, 2020) where the best result obtained with BERT was **0.5958** (binary) and **0.5105** (proportional).

The transformer models implemented in this section are the NorBert (2022) and the MultiBERT (Multilingual Cased) (Devlin and Petrov, 2019). As the name suggests, NorBERT (ID 216) is a BERT model trained on Norwegian data[5]. MultiBERT is a large multilingual model containing 104 languages. Both BERT models were implemented with the *AutoModel* for activating the model and *AutoTokenizer* for tokenizing the sentences from the *transformers* package in PyTorch. On top of the BERT model, a linear layer in order to project the output into the right number of classes.

3.4 Character-level information

Traditionally, we built Neural Language Models using words in a sentence or paragraph or even document. And this indeed provides us with solid performance on different NLP tasks (Darwish, 2013; Hakan Demir, 2014). However, using words to train the model will generate a huge vocabulary and may hinder the improvement of models due to their capacity for handling unknown words, punctuation, and other document structures. Furthermore, this may cause the model to require more time and resources to train. Nevertheless, the character-based models overcome these drawbacks and provide a promising probability. A Hidden Markov Model (HMM) with character-level information could reduce the data sparsity problem, which is inherently present in word-level information, and get 25% error reduction compared with the same model without character information

[3] http://wiki.nlpl.eu/Vectors/norlm/norelmo

[4] https://github.com/allenai/allennlp

[5] http://wiki.nlpl.eu/Vectors/norlm/norbert

(Dan Klein, 2003). Previous studies have proved that the CharacterBERT has better performance than the BERT on different medical domain tasks (Hicham El Boukkouri, 2020).

There are mainly two ways of using character-level information for NLP tasks, which vary based on the nature of the task. For example, Yoon Kim (2015) used the characters to generate a word representation for each token in one sentence, where they employed CNN for word representation. On the other hand, Xiang Zhang (2015) used character information which was not mapped to words first, where they also used CNN but were directly mapped to sentiments and token categories.

In our implementation, we chose the first way of using character information and tried to add the character level information to our word-based deep neural networks. The word character was first encoded with the generated alphabet based on all three data sets, and the character information was extracted using random initialized embedding and applying a 1D convolution layer. After getting the character information, we just concatenate the character features with the features from other methods such as RNNs, BERTs, and ELMo and feed them together to the classification layer.

3.5 Label encoding

There is quite a lot of research effort on NER or TSA, especially generating the state-of-the-art models. However, little research has studied the effects of different annotation schemes on NER or TSA tasks. Based on the authors' knowledge, there is no adequate work that investigates the impact and implications of utilizing multi-annotation systems on Norwegian NER or TSA tasks in the literature. To fill this gap, we will look into the effects of alternative annotation systems on the Norwegian TSA problem. **B**egin **I**nside **O**utside (BIO) of entities are popularly tagging format used for token tagging in chunking tasks like named entity recognition. Unit and last tags were added in BIOUL format to further distinguish the end of tokens and mid-entity tokens, and represent the single token entity. Basically, it may be unnecessary for regular named-entity recognition to distinguish the end of entity token labels from the single entity label. However, it could make a difference when using Conditional random fields (CRFs).

In this project, we created a Python script that transfers the original dataset, which is BIO encoded, to the BIOUL encoded `conll` data file. Then the data file was used in the same way as the original data, with some minor adjustment of the code. The table 4 illustrates an example of the same sentence as table 3 yet with the BIOUL encoding scheme. The new dataset contains the following nine classes

(1) **O**: outside
(2) **B-targ-Positive**: beginning of a token with positive sentiment
(3) **I-targ-Positive**: inside of a token with positive sentiment
(4) **L-TARG-POSITIVE**: last of a token with positive sentiment
(5) **U-TARG-POSITIVE**: single token without L and I with positive sentiment
(6) Same for the negative sentiments for points 2-5.

Han	U-targ-Neg
redda	O
den	O
perfekte	O
hustrue-skuespilleren	O
Bonnie	B-targ-Pos
Bedelia	L-targ-Pos
og	O
blødde	O
overfladisk	O
.	O

Table 4: Same example as table 3 but with BIOUL encoding. Note that this sample contains U-targ-Neg and L-targ-Pos.

3.6 Evaluation

The models were evaluated by *proportional* F_1 and *binary* F_1 metrics. For the proportional, precision is defined as the ratio of overlap with the projected span, whereas recall is defined as the ratio of overlap with the gold span, which reduces to token-level F_1. Any overlapping anticipated and gold span is considered correct by binary. Based on this, we may anticipate that the model will provide better binary F_1 than proportional F_1. a

4 Training

The training of the models was compute-intensive and time consuming. We trained the models on Saga and the Google Colab with a seed number of 5550.

4.1 Tuning parameters

In order to maximize the performance of the model, we tuned the hyper-parameters. The list below shows which parameters were tuned for the different architectures:

1) **LSTM & GRU**: grid search of hidden dimension, amount of dropout between layers, amount dropout in RNN model (except for 1 layer model), whether they were bidirectional or not, number of layers in the RNN and whether character-level information is included or not.

2) **ELMo**: tuning of the RNN model on top of the embedding layer. Almost the same parameters as above; hidden dimension, number of RNN layers, and the amount of dropout.

3) **BERT**: testing BERT and whether character-level information improves the performance or not.

4) **BIOUL encoding**: The models with the best performance were retrained with the BIOUL dataset.

In total, we trained 96 RNN models, 24 ELMo models, and 8 BERT models.

5 Results

The hyper-parameters of each models are listed in the Appendix section.

5.1 Best results with BIO encoding

The best performances of the models are found in table 5. It is clear that NorBERT won among the rest of the models, with **0.644** for binary F1 and **0.513** for proportional F1. And it is noted that all pre-trained models perform better than RNNs. The ELMo and the Multilingual BERT have quite similar results.

5.2 Character-level information

The best performances of the models that were trained with added character-level features are found in table 6. The results show that the added character information makes the models even worse for GRU and LSTM and has little influence on pre-trained models.

Architecture	Binary F_1	Proportional F_1
Baseline	0.328	0.157
GRU	0.383	0.201
LSTM	0.428	0.257
ELMo	0.597	0.476
NorBERT	**0.644**	**0.513**
MultiBERT	0.597	0.440

Table 5: The best performance of the each architectures with BIO encoding. Baseline refers to the BiLSTM with default value. GRU, LSTM, ELMo, NorBERT and MultiBERT refer to the combination of different hyperparameters giving best performance.

Architecture	Binary F_1	Proportional F_1
GRU	0.221	0.134
LSTM	0.398	0.199
NorBERT	**0.642**	**0.516**
MultiBERT	0.593	0.435

Table 6: The best performance of the architectures with included character-level information with BIO encoding.

5.3 Label encoding

The dataset with BIOUL encoding was used to retrain the NorBERT and ELMo models. The result from this is in the table 7. Both scores are a little bit lower than the model on BIO encoded data.

6 Discussion

The fine tuning of hyperparameters improved the baseline model from **0.328** to **0.428** for binary and **0.157** to **0.257** for proportional F_1, as shown in table 5. And similar results were also found in GRU models, but to a smaller extent. However, the fine-tuned version of the LSTM and GRU models do not produce very promising results for TSA on the $NoReC_{fine}$ dataset, with performance still falling below 50%.

Architecture	Binary F_1	Proportional F_1
ELMo	0.582	0.469
NorBERT	**0.614**	**0.483**

Table 7: The performance of NorBERT and ELMo with BIOUL encoding.

Architecture	Time taken/epoch
ELMo	$\sim$ 100s
NorBERT	$\sim$ 110s

Table 8: The time taken for training 1 epoch of NorBERT and ELMo at GPU.

The pretrained models, on the other hand, gave quite promising results. In the table 5, we can see that the NorBERT model has a significant improvement compared with the baseline. This was actually expected since BERT models are the most dominant architectures in the NLP world for the moment. NorBERT is based primarily on Norwegian text from the ground up, which may explain why it outperforms the other pretrained model. This is in line with the research efforts of Jeremy Barnes (2021) and Rønningstad (2020) in this field, where they have also built BERT model on TSA in $NoReC_{fine}$ dataset and achieved the best scores of **0.5958** for binary F_1, **0.5105** for proportional F_1 and **0.6271** for binary F_1, **0.4970** for proportional F_1, respectively. Our score is slightly higher, probably due to the random seed. It is interesting that Rønningstad got his best performance with the Multi-lingual BERT model, while Barnes, like us, obtained the best score with NorBERT.

Another interesting finding was that the Norwegian Elmo model performed better than the MultiBERT, but it could not beat the NorBERT. However, the ELMo is less time-consuming than the NorBERT while training (table 8). With regard to the training time for RNNs, it takes only seconds for each epoch, which is not comparable to the pre-trained models.

The addition of character-level information to the models does not seem to improve the performances, and even makes the RNNs worse. The implementation of character features is a bit naive. We just used the random initialized embedding for the characters of words and did not include any further processing for character features. This could be the possible reason that the extra features from character level does not improve the performance of models. The pre-trained models seem more stable when adding character features, compared with the RNNs.

The resulting scores for NorBERT and ELMo on BIOUL encoding data are similar to those on BIO encoding data. Since we just tested NorBERT and ELMo on BIOUL encoding data, it is unknown whether the RNNs on BIOUL and BIO encoding data vary a lot. On the other hand, Alshammari and Alanazi (2021), and Aasmoe (2019) compared the effects of different encodings for NER on Arabic and Norwegian dataset, and found a better performance on the BIOUL encoding scheme than the BIO encoding, but in this case, we obtained the opposite.

6.1 Conclusion

In this project, we performed targeted sentiment analysis on the $NoReC_{fine}$ dataset. We implemented various neural architectures; GRU, LSTM, ELMo, NorBERT, and MultiBERT. These models were trained with two types of encodings; the BIO encoding from the dataset and the BIOUL encoding. We have fine-tuned the models in order to find the best performance. In our case, we found out that the NorBERT performed best and the Norwegian ELMo performed a bit behind the NorBERT. However, we also confirmed the *performance vs. time* when training the NorBERT and ELMo. ELMo is less time-consuming, but its performance is not as good as NorBERT's. Another interesting finding was that the pre-trained models perform best with BIO encoding and a bit worse with BIOUL.

7 Future work

This section outlines a number of potential directions for this project that were not included in the original scope.

7.1 Error analysis

The NorBERT model obtained almost the same result with and without including the character-level information. The performance did not differ too much when the NorBERT model was trained with the BIOUL encoding. In total, these factors did not impact the result of the model at all, which seems a bit strange. Therefore, an error analysis would be interesting in this case, which could help explore the decisions of the classifier in more detail and try to find out why the classifier is obtaining the same result for different encoding types and with/without including character-level information. More specifically, these questions would be interesting

(1) Is there any type of sentences in the original dataset that NorBERT cannot correctly classify? What kinds of sentences does it struggle with?

(2) How does the NorBERT handle such sentences with BIOUL encoding?

7.2 Character-level information

The addition of character information did not give us better results for all the architectures in our implementation. The next study could use character features in the same way that previous research has shown to improve model performance. One example is Hicham El Boukkouri (2020), where he used the character feature by firstly applying two non-linear Highway layers, which are basically relatively deep feed-forward neural networks, before projecting to embedding and processing with CNN. It is interesting to see if the addition of character features would have the same improvement for TSA.

7.3 Other types of encoding

Due to the limited time, we did not implement the performance of more different encoding schemes on all of the different architectures. In this paper, we only experimented with the encoding types of BIO and BIOUL, and obtained some interesting results, especially with BIO. Other interesting encoding types like IO, BIOU, or BIOL are also worth experimenting with and could provide a much more comprehensive understanding of the effects of different encoding schemes.

References

Tobias Langø Aasmoe. 2019. Named entity recognition for norwegian - experiments on the norne corpus.

Nasser Alshammari and Saad Alanazi. 2021. The impact of using different annotation schemes on named entity recognition. *Egyptian Informatics Journal*, 22(3):295–302.

Caiming Xiong Richard Socher Bryan McCann, James Bradbury. 2017. Learned in translation: Contextualized word vectors.

Huy Nguyen Christopher D. Manning Dan Klein, Joseph Smarr. 2003. Named entity recognition with character-level models.

Kareem Darwish. 2013. Named entity recognition using cross-lingual resources: Arabic as an example. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1558–1567, Sofia, Bulgaria. Association for Computational Linguistics.

Jacob Devlin and Slav Petrov. 2019. Multilingual bert.

ELMo. 2022. Elmo — Wikipedia, the free encyclopedia. [Online; accessed May-2022].

Arzucan Ozgur Hakan Demir. 2014. Improving named entity recognition for morphologically rich languages using word.

Thomas Lavergne Hiroshi Noji Pierre Zweigenbaum Junichi Tsujii Hicham El Boukkouri, Olivier Ferret. 2020. Characterbert: Reconciling elmo and bert for word-level open-vocabulary representations from characters.

Vinit Ravishankar Erik Velldal Lilja Øvrelid Jeremy Barnes, Andrey Kutuzov. 2021. Proceedings of the third in5550 workshop on neural language processing (wnnlp 2021). page 20.

Jeremy Barnes Lilja Øvrelid, Petter Mæhlum and Erik Velldal. 2020. A fine-grained sentiment dataset for norwegian. pages 5025–5033.

NorBert. 2022. Norbert — Nordic language processing laboratory, the free encyclopedia. [Online; accessed May-2022].

Ido Dagan Oren Melamud, Jacob Goldberger. 2016. context2vec: Learning generic context embedding with bidirectional lstm. pages 51–61.

Egil Rønningstad. 2020. Targeted sentiment analysis for norwegian text.

Yann LeCun Xiang Zhang. 2015. Text understanding from scratch.

David Sontag Alexander M. Rush Yoon Kim, Yacine Jernite. 2015. Character-aware neural language models.

Haiyun Peng Yukun Ma and Erik Cambria. 2018. Targeted aspect-based sentiment analysis via embedding commonsense knowledge into an attentive lstm.

Haiyun Peng Yukun Ma and Erik Cambria. 2019. Applying deep learning approach to targeted aspect-based sentiment analysis for restaurant domain. pages 206–211.

Meishan Zhang, Yue Zhang, and Duy-Tin Vo. 2016. Gated neural networks for targeted sentiment analysis.

A Appendix

A.1 Parameters for models in table [5]

- GRU model:
  - NUM_LAYERS=1
  - HIDDEN_DIM=500
  - WORD_DROPOUT=0.6
  - RNN_DROPOUT=0.2
  - EPOCHS=30
  - BIDIRECTIONAL=False
- LSTM model:
  - NUM_LAYERS=1
  - HIDDEN_DIM=500
  - WORD_DROPOUT=0.6
  - RNN_DROPOUT=0.2
  - EPOCHS=30
  - BIDIRECTIONAL=False
- ELMo:
  - CLASSIFIER =’gru’
  - DROPOUT=0.5
  - EPOCHS=20
  - HIDDEN_DIM=300,
  - NUM_LAYERS=3
- BERT:
  - EPOCHS=50
  - FREEZE=False,
  - NUM_LAYERS=1
  - HIDDEN_DIM=100

A.2 Parameters for models in table [6]

- GRU model:
  - NUM_LAYERS=2
  - CHAR_EMBEDDING_DIM=30
  - CHAR_HIDDEN_DIM=50
  - WORD_DROPOUT=0.5
  - RNN_DROPOUT=0.2
  - EPOCHS=30
  - BIDIRECTIONAL=True
- LSTM model:
  - NUM_LAYERS=2
  - CHAR_EMBEDDING_DIM=30
  - CHAR_HIDDEN_DIM=50
  - WORD_DROPOUT=0.5
  - RNN_DROPOUT=0.2
  - EPOCHS=30
  - BIDIRECTIONAL=True
- BERT:
  - EPOCHS=50
  - FREEZE=False,
  - NUM_LAYERS=1
  - HIDDEN_DIM=100

A.3 Parameters for models in table [7]

- ELMo:
  - CLASSIFIER =’gru’
  - DROPOUT=0.5
  - EPOCHS=20
  - HIDDEN_DIM=300,
  - NUM_LAYERS=3
- BERT:
  - EPOCHS=50
  - FREEZE=False,
  - NUM_LAYERS=1
  - HIDDEN_DIM=100