N-gram and neural models for Uralic language identification: NRC at VarDial 2021

Bernier-Colborne, Gabriel; Léger, Serge; Goutte, Cyril

Publisher's version / Version de l'éditeur:

Proceedings of the Eighth VarDial Workshop on NLP for Similar Languages, Varieties and Dialects, pp. 128-134, 2021-04-20

NRC Publications Archive Record / Notice des Archives des publications du CNRC :
https://nrc-publications.canada.ca/eng/view/object/?id=ff9f3cc0-73cd-4026-bcf7-3b60e247330d
https://publications-cnrc.canada.ca/fra/voir/objet/?id=ff9f3cc0-73cd-4026-bcf7-3b60e247330d

N-gram and Neural Models for Uralic Language Identification: NRC at VarDial 2021

Gabriel Bernier-Colborne, Serge Léger, Cyril Goutte
National Research Council Canada
{Gabriel.Bernier-Colborne, Serge.Leger, Cyril.Goutte}@nrc-cnrc.gc.ca

Abstract

We describe the systems developed by the National Research Council Canada for the Uralic language identification shared task at the 2021 VarDial evaluation campaign. We evaluated two different approaches to this task: a probabilistic classifier exploiting only character 5-grams as features, and a character-based neural network pre-trained through self-supervision, then fine-tuned on the language identification task. The former method turned out to perform better, which casts doubt on the usefulness of deep learning methods for language identification, where they have yet to convincingly and consistently outperform simpler and less costly classification algorithms exploiting n-gram features.

1 Introduction

The goal of the Uralic language identification (ULI) shared task at VarDial 2021 (Chakravarthi et al., 2021) was to identify and discriminate 29 Uralic language varieties, among a total of 178 language varieties from various families, given a short text (typically a sentence). This was a re-run of the ULI task at VarDial 2020 (Găman et al., 2020).

We experimented with two different approaches to the ULI task. The first is a probabilistic classifier that exploits only character 5-grams as features. The second is a deep learning approach based on character embeddings and a transformer network (Vaswani et al., 2017), which is first pre-trained through self-supervision, then fine-tuned on the ULI task, in a similar fashion to BERT (Devlin et al., 2019).

This second approach is essentially an improved version of the one developed by the NRC team for the first run of the ULI task (Bernier-Colborne and Goutte, 2020). By improving the sampling functions used to sample training and evaluation data, and making a few other small changes to the model, we were able to achieve much better results. However, our best results were achieved using the simpler approach, based on character 5-grams.

In this paper, we explain these two approaches to language identification and compare the results obtained on the ULI task.

2 Data and Task Definition

The ULI data and task are described by Jauhiainen et al. (2020). The goal of the task is to identify the language of a short text, typically a single sentence. If more than one language is used (e.g. code switching), the main language must be identified.

This was a closed task, so the only data that could be used was the training set provided. The data contain both relevant and non-relevant languages. The 29 relevant languages are part of the Uralic group of languages, spoken mainly in northern Eurasia. Some are very under-resourced, and some are already extinct. Besides these, there are 149 non-relevant languages, belonging to various language families. These include the three largest Uralic languages, i.e. Estonian, Finnish, and Hungarian. In total, the data covers 178 language varieties, which is the highest number covered so far in a language identification shared task.

The training set contains 646,043 examples for the relevant languages and 63,772,445 examples for the non-relevant ones. So there is about 100 times more data for the non-relevant languages, and there are 5 times more non-relevant languages than relevant ones, therefore the number of examples is about 20 times greater for non-relevant languages, on average. Both parts of the training set are highly unbalanced. The class frequencies of relevant languages range from 19 to 214,225, and those of non-relevant ones range from 10,000 to 3,000,000.

The task was divided into three tracks, the difference being the way in which the evaluation metric

Proceedings of the 8th VarDial Workshop on NLP for Similar Languages, Varieties and Dialects, pages 128–134
April 20, 2021 ©2021 Association for Computational Linguistics

is computed. For tracks 1 and 2, the evaluation metrics are the macro-averaged F1 and micro-averaged F1 respectively, and it only considers examples where either the predicted label or the true label is a relevant language. For track 3, the metric is macro-averaged F1, computed over all examples.

Some of the challenges of this task were highlighted by Bernier-Colborne and Goutte (2020) and Jauhiainen et al. (2020). These include: the presence of low-resource and closely-related language varieties; the large size of the training set; the large and complex class imbalances in the training set; and the absence of an official development set, which, while a valid design choice for this competition, added a degree of complexity.

3 Models

We evaluated two approaches to this task, which we describe in this section.

3.1 Probcat

The first approach employs a probabilistic classifier (Gaussier et al., 2002), that we call Probcat, which we trained using character 5-grams as features. The classifier is similar to multinomial Naive Bayes except that it does not assume that all n-grams in a given text are generated from a single class. It has been used in the past to obtain state-of-the-art results on language identification tasks (Goutte and Léger, 2016) . For more details on this classification algorithm, refer to Goutte et al. (2014, Sec. 2.2).

For reference, running inference on the official test, which contains over 1.5 million examples, takes about four and a half hours using four Intel Xeon CPUs @ 2.6 GHz, and could be reduced by using more CPUs. Training takes about five and a half hours. Memory requirement is below 32 GB.

3.2 Transformer

The second approach is a deep learning approach, which the NRC team has previously applied, with very different levels of success, to Cuneiform language identification (Bernier-Colborne et al., 2019) and Uralic language identification (Bernier-Colborne and Goutte, 2020). In the former case, it produced a winning submission, whereas in the latter, it was well under the baseline, because of a flaw in the methods used to sample training and evaluation data.

The model is a deep neural network which takes sequences of characters as input. Characters are embedded and fed through a stack of bidirectional transformers (Vaswani et al., 2017), which encodes the sequence. The output of this encoder is a sequence of hidden state vectors (one per input character), which is then fed to various output heads (or modules) during training.

The model is trained in two stages: self-supervised pre-training on a masked language modeling (MLM) task (Devlin et al., 2019), followed by supervised fine-tuning on the target task, i.e. language identification. For MLM, the objective is to predict one or more randomly chosen characters in the input sequence, which are replaced with a special masking symbol before the embedding stage. This produces a model that can predict characters in context for any of the languages used for training, and must therefore have learned some of the specific surface regularities of each. The output head for this task is a softmax over the vocabulary (or alphabet), which takes as input the encoding (i.e. final hidden state) of a masked character, and the loss is cross-entropy.

After pre-training, we fine-tune for language identification by keeping the pre-trained encoder, discarding the MLM head, and replacing it with a head containing: an average pooling layer, that averages the final hidden states of the encoder to produce a fixed-length encoding of the sentence (Reimers and Gurevych, 2019); followed by a relu activation; and finally a softmax over the 178 languages. Again, the loss is cross-entropy.

The vocabulary (or alphabet) contains every character that appears more than once in the training portion of the train/dev/test split we created (see Section 4.1). Characters that are not in this vocab are replaced with a special symbol reserved for unknown characters, before the encoding stage.

The hyperparameter settings we used largely follow the recommendations of Devlin et al. (2019), except that we use fewer layers than their base architecture (to reduce training time and memory requirements):

- Nb transformer layers: 8
- Nb attention heads: 12
- Hidden layer size: 768
- Feed forward/filter size: 3072
- Hidden activation: gelu

- Dropout probability: 0.1
- Optimizer: Adam
- Learning rate (pre-training): 1e-4
- Nb steps (pre-training): 1M
- Warmup steps (pre-training): 10K
- Batch size (pre-training): 64
- Maximum input length (pre-training): 128
- Learning rate (fine-tuning): 1e-5
- Batch size (fine-tuning): 32
- Maximum input length (fine-tuning): 256

During pre-training, examples were sampled from our training set using the frequency-based sampling function described by Bernier-Colborne and Goutte (2020). We refer the reader to this paper for the full equations. In essence, we compute the relative frequencies of relevant and non-relevant languages separately, damp each of the two resulting distributions using exponent α, multiply the distribution of relevant languages by coefficient γ, then combine the two distributions and re-normalize. We arbitrarily set $\alpha = 1$ and $\gamma = 1$, which means that relevant and non-relevant examples are sampled in approximately equal proportion, so the relevant languages will end up being sampled about 5 times more frequently than non-relevant ones, on average, as there are about 5 times fewer relevant languages.

During fine-tuning, we experimented with various functions to sample the labeled training examples from the training set, including this frequency-based function. The other functions we experimented with were:

- Class-wise uniform sampling
- Accuracy-based sampling using a dynamic estimate of class-wise accuracy. We tested two different functions to estimate this, both based on the same score, which is the class-wise, one-vs-rest binary accuracy on the dev set. If we call the set of class-wise scores m, then the two functions we tested to compute sampling probabilities can be formulated as follows:

$$p_{\text{inv}}(m_i) = \frac{1 - m_i}{\sum_j (1 - m_j)}$$

$$p_{\text{rank}}(m_i) = \frac{\text{rank}(m_i, m)}{\sum_j \text{rank}(m_j, m)}$$

where $\text{rank}(m_i, m)$ returns the rank of m_i.

During fine-tuning, the maximum input length was increased to 256; the extra position embeddings are still randomly initialized at this point, and are learned during fine-tuning. To reduce useless computation, rather than padding or truncating all sequences in a given batch to the maximum length of 256, we pad or truncate them to the length of the longest sequence in the batch. This change reduced computation time significantly with respect to our previous implementation, as the vast majority (over 99%) of examples in the ULI training data are shorter than 256 characters.

Our code, which exploits the Transformers library by HuggingFace (Wolf et al., 2020), is available is at https://www.github.com/gbcolborne/vardial2021.

All experiments were conducted on a single GPU with 12 GB of memory. Inference on the official test set takes about 70 minutes using a v100. Pre-training took about 10 days, and we fine-tuned for about 3 days.

4 Experiments

To evaluate and optimize our two approaches, we first had to create a development set, as none was provided for this shared task, as mentioned in Section 2.

4.1 Making a Development Set

To sample our development sets, we used the frequency-based sampling function described in Section 3.2, with $\alpha = 1$ and $\gamma = 1$, to sample a 'dev' set containing 20,000 examples and a 'dev-test' set containing 100,000 examples. The idea was to use the dev set to tune the hyperparameters of the models, then use the dev-test set to get an unbiased estimate of the accuracy of the fully tuned models. However, due to time constraints, in the case of the transformer model, we ended up adding the dev-test set to the training set, and using the dev set for model selection. We will come back to this in the following section.

4.2 Model Selection

To optimize Probcat, we compared 4 different n-gram lengths, from 2 to 5. The models were trained on our training set and evaluated on both our development sets. Results showed that 5-grams produced the best results, as shown in Table 1. Note that after the official evaluation, we tried combining various n-gram lengths, but did not observe any improvements on our development sets.

n	Track 1		Track 2		Track 3	
	Dev	Test	Dev	Test	Dev	Test
2	0.760	0.763	0.769	0.774	0.766	0.743
3	0.917	0.844	0.941	0.936	0.912	0.890
4	0.960	0.916	0.973	0.970	0.939	0.929
5	**0.963**	**0.936**	**0.978**	**0.976**	**0.951**	**0.943**

Table 1: F-scores of Probcat on dev and dev-test sets, with respect to n-gram length (n).

We made a single submission using this approach, using only 5-grams as features. The entire training set was used to train the classifier for this submission.

As for the transformer model, we conducted various experiments involving: optimizing the architecture (e.g. replacing the tanh activation in the pooling layer with a relu activation); developing better sampling functions for training; and tuning a few hyperparameters, such as the learning rate and batch size. We will not present the full results of these experiments here, because of their ad hoc nature.

We ended up making a first set of two submissions, and a final set of four submissions. As the results of the latter were better, we will not go into the model selection tests that led to the first set of submissions.

For our final model selection experiment, we decided to add the dev-test set to the training set (for lack of time to re-train models on the entire training set), and rely on the dev set evaluation to select models. Note that, because of this reason, the results of these tests are not directly comparable to the dev scores of Probcat.

We focused on the sampling function used to fine-tune the model, and compared the functions described in Section 3.2:

- Frequency-based sampling, with a few different values of α and γ, i.e. $(\alpha = 1, \gamma = 1)$, $(\alpha = 0.75, \gamma = 1)$, and $(\alpha = 0.75, \gamma = 0.5)$.
- Uniform sampling.
- Accuracy-based sampling, using either p_{inv} or p_{rank} to convert class-wise dev-scores to sampling probabilities.

We compared a couple different learning rates (i.e. 3e-6 and 1e-5). Models were fine-tuned for a maximum of 820K steps, and validated every 20K steps, with early stopping on the dev set, using either the track 1 or track 3 score as stopping criterion.

Sampling	Track 1	Track 2	Track 3
Frequency	**0.977**	**0.989**	**0.978**
Uniform	0.973	0.986	0.969
Acc. (p_{inv})	0.974	0.988	0.977
Acc. (p_{rank})	0.975	0.987	0.974

Table 2: Best F-scores of transformer model on dev set, with respect to sampling function.

The results, summarized in Table 2, suggested frequency-based sampling worked best, but all sampling functions achieved high scores, with accuracy-based sampling working slightly better than uniform sampling. The optimal learning rate was 1e-5.

We submitted four runs, two of which were tuned (in terms of the sampling function and early stopping) to track 1, and two of which were tuned to track 3:

- Track 1, run 1: frequency-based sampling (with $\alpha = 0.75, \gamma = 0.5$), early stopped for track 1.
- Track 1, run 2: accuracy-based sampling (with p_{rank}), early stopped for track 1.
- Track 3, run 1: frequency-based sampling (with $\alpha = 0.75, \gamma = 0.5$), early stopped for track 3.
- Track 3, run 2: accuracy-based sampling (with p_{inv}), early stopped for track 3.

5 Official Results

The scores of our best runs on the official test set are shown in Table 3. The baseline scores were computed using the HeLI method (Jauhiainen et al., 2017). Probcat was the only system to beat the baseline on track 2, and one of two systems that beat the baseline on track 1 (along with an ensemble of SVM and naive Bayes models exploiting 3-, 4-, and 5-grams, which scored 0.809 on track 1, but only 0.593 on track 2 according to the leaderboard[1]

[1] http://urn.fi/urn:nbn:fi:lb-2020102201

at the time of writing). The baseline on track 3 has yet to be beaten.

Our results using the transformer model are competitive on tracks 2 and 3, but less so on track 1, which focuses on the macro-averaged F-score on relevant languages.

System	Track 1	Track 2	Track 3
Baseline	0.800	0.963	**0.925**
Probcat	**0.814**	**0.967**	0.908
Transformer	0.743	0.953	0.904

Table 3: F-scores of our best runs and the baseline system (i.e. HeLI) on the offical test set.

The scores of our final 4 runs of the transformer model are shown in Table 4.

Run	Track 1	Track 2	Track 3
Track 1, run 1	0.718	0.928	0.898
Track 1, run 2	0.737	**0.953**	**0.904**
Track 3, run 1	**0.743**	0.928	0.899
Track 3, run 2	0.615	0.830	0.784

Table 4: F-scores of our 4 final runs of the transformer model on the offical test set.

These results suggest our development sets were not very good estimators for model selection, as a model that we (slightly) tuned for track 1 was best on tracks 2 and 3, whereas a model we tuned for track 3 was best on track 1. If we check how many training steps each of the four models did before early stopping, we see that the model that stopped earliest produced the poorest results on the test set:

- Track 1, run 1: step 649375
- Track 1, run 2: step 651875
- Track 3, run 1: step 726875
- Track 3, run 2: step 590625

We also see that the model that stopped the latest produced the best results on track 3; since the batch size was 32, that model observed about 23 million samples from the training set (including duplicate samples from rare classes), which is less than the total number of available training samples. Given these observations, it seems likely that we could have obtained better results by fine-tuning longer. Other ways of improving the accuracy of this model include: trying multiple random initializations; exploring the hyperparameter space more extensively or efficiently; fine-tuning on the complete training set; or adapting the model to the unlabeled test examples (e.g. through self-supervised MLM or self-training on the language identification task).

We could not conduct extensive error analysis because we did not have access to the test labels at the time of writing, but we did inspect the predictions of our systems to try to gain insights on their behaviour or the properties of the data.

If we look at the cumulative predicted frequency of the 29 relevant languages, we find that Probcat predicted a relevant language for 21,051 or 1.4% of the 1,510,315 test cases. Since Probcat achieved 0.967 on track 2, this must be very close to the actual number of relevant examples in the test set. Thus, the distribution of relevant vs. non-relevant examples is very different than the one we assumed when we constructed our dev set. Furthermore, the worst of our four transformer runs seems to have over-detected relevant examples, as it predicted a relevant language for 27,901 test cases.

If we look at the top off-diagonal values in the confusion matrix between Probcat and our best transformer run (i.e. track 1, run 2, which performed best overall) for all 178 languages, assuming Probcat's predictions as ground truth (as it performed better), we find that the 10 most frequently confused pairs (Probcat prediction, transformer prediction) are:

1. Indonesian, Sundanese (10110)
2. Tatar, Bashkir (9897)
3. Indonesian, Wu Chinese (6184)
4. Indonesian, Standard Malay (5957)
5. Javanese, Sundanese (5550)
6. Croatian, Bosnian (5312)
7. Mandarin Chinese, Wu Chinese (4453)
8. Japanese, Wu Chinese (4136)
9. Iranian Persian, Pushto (3806)
10. Italian, Lombard (2574)

These contain pairs that are notoriously hard to distinguish, e.g. Croatian and Bosnian.

If we restrict this analysis to pairs involving a relevant language, we find that the top 10 most confused pairs are:

1. Standard Estonian, Võro (467)
2. Võro, Standard Estonian (388)
3. Standard Estonian, Veps (86)
4. Finnish, Tornedalen Finnish (51)
5. Ludian, Veps (44)
6. Ludian, Livvi (43)
7. Tornedalen Finnish, Finnish (35)
8. Russian, Eastern and Meadow Mari (32)
9. Haitian, Veps (31)
10. Finnish, Võro (15)

Note that several of these involve a non-relevant Uralic language, i.e. Standard Estonian or Finnish, which raises the question as to whether the Standard Estonian and Finnish training corpora contain noisy examples actually belonging to a relevant, low-resource language variety.

We manually inspected some of the test cases for which Probcat predicted a language that we understand, i.e. English, and looked for potential sources of errors. Without access to the gold labels, we could not glean much, but we did observe examples where English is used within sentences that are mainly in another language:

- Kanta horien artean ezagunenak "Cocaine", "After Midnight", "Call Me the Breeze", "Travelling Light" eta "Sensitive Kind" dira.
- 外国人以为皮蛋要腌交关年，所以叫渠Century egg（世纪蛋）。

And examples that are not in Standard English, but a closely related language or variety:

- The GT wis sauld alangside the GTi for a few months, but wis eventually phased oot.
- The population increased frae 76,254 inhabitants (1992 census) tae 77,566 (2001 census), an increase o 1.7 %.
- Ships o the Hudson's Bay Company wur regular visitors, as wur whalin fleets.

We plan on conducting a more extensive error analysis once the gold labels of the test set are made available.

6 Related Work

Thorough surveys of research on language identification are provided by Jauhiainen et al. (2019) and Zampieri et al. (2020). Language identification is one of the few tasks in natural language processing where deep learning methods have yet to provide convincing gains in accuracy or robustness. Simpler classifiers based on character n-grams continue to provide state-of-the-art results on many benchmarks. The winning submission by the NRC team to the Cuneiform language identification task at VarDial 2019 was the first time a neural system was ranked first on a language identification shared task (Zampieri et al., 2019). That task involved 7 language varieties, and less complex class imbalances. The results of this Uralic language identification shared task cast doubt on whether a character-based deep neural network can advance the state of the art in settings more representative of real-world applications of language identification, such as crawling the web to automatically compile monolingual corpora for low-resource languages.

7 Conclusion

For the Uralic language identification shared task at the 2021 VarDial evaluation campaign, the NRC team evaluated two different approaches: a probabilistic classifier exploiting only character 5-grams as features, and a character-based neural network pre-trained through self-supervision, then fine-tuned on the language identification task. The former method ended up being ranked first on two of the three tracks, and outperformed our neural approach on all three tracks. We do not exclude the possibility that deep learning approaches could improve accuracy or robustness on this task, but the results we were able to obtain within the limited time constraints of this shared task suggest that the simpler, n-gram based approach is still a very strong baseline.

Acknowledgments

We thank the organizers for their work developing and running this shared task.

References

Gabriel Bernier-Colborne and Cyril Goutte. 2020. Challenges in neural language identification: NRC at VarDial 2020. In *Proceedings of the 7th Workshop on NLP for Similar Languages, Varieties and*

Dialects, pages 273–282, Barcelona, Spain (Online). International Committee on Computational Linguistics (ICCL).

Gabriel Bernier-Colborne, Cyril Goutte, and Serge Léger. 2019. Improving cuneiform language identification with BERT. In *Proceedings of the Sixth Workshop on NLP for Similar Languages, Varieties and Dialects*, pages 17–25, Ann Arbor, Michigan. Association for Computational Linguistics.

Bharathi Raja Chakravarthi, Mihaela Găman, Radu Tudor Ionescu, Heidi Jauhiainen, Tommi Jauhiainen, Krister Lindén, Nikola Ljubešić, Niko Partanen, Ruba Priyadharshini, Christoph Purschke, Eswari Rajagopal, Yves Scherrer, and Marcos Zampieri. 2021. Findings of the VarDial Evaluation Campaign 2021. In *Proceedings of the Eighth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL*, pages 4171–4186.

Eric Gaussier, Cyril Goutte, Kris Popat, and Francine Chen. 2002. A hierarchical model for clustering and categorising documents. In *Proceedings of the 24th BCS-IRSG European Colloquium on IR Research: Advances in Information Retrieval*, pages 229–247. Springer-Verlag.

Cyril Goutte and Serge Léger. 2016. Advances in Ngram-based Discrimination of Similar Languages. In *Proceedings of the Third Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial3)*, pages 178–184, Osaka, Japan.

Cyril Goutte, Serge Léger, and Marine Carpuat. 2014. The NRC system for discriminating similar languages. In *Proceedings of the First Workshop on Applying NLP Tools to Similar Languages, Varieties and Dialects (VarDial)*, pages 139–145, Dublin, Ireland.

Mihaela Găman, Dirk Hovy, Radu Tudor Ionescu, Heidi Jauhiainen, Tommi Jauhiainen, Krister Lindén, Nikola Ljubešić, Niko Partanen, Christoph Purschke, Yves Scherrer, and Marcos Zampieri. 2020. A Report on the VarDial Evaluation Campaign 2020. In *Proceedings of the Seventh Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*.

Tommi Jauhiainen, Heidi Jauhiainen, Niko Partanen, and Krister Lindén. 2020. Uralic Language Identification (ULI) 2020 shared task dataset and the Wanca 2017 corpus. In *Proceedings of the Seventh Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*, pages 688–698.

Tommi Jauhiainen, Krister Lindén, and Heidi Jauhiainen. 2017. Evaluating HeLI with non-linear mappings. In *Proceedings of the Fourth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*, pages 102–108, Valencia, Spain.

Tommi Jauhiainen, Marco Lui, Marcos Zampieri, Timothy Baldwin, and Krister Lindén. 2019. Automatic Language Identification in Texts: A Survey. *Journal of Artificial Intelligence Research*, 65:675–782.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi 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 M. Rush. 2020. HuggingFace's Transformers: State-of-the-art natural language processing.

Marcos Zampieri, Shervin Malmasi, Yves Scherrer, Tanja Samardžić, Francis Tyers, Miikka Silfverberg, Natalia Klyueva, Tung-Le Pan, Chu-Ren Huang, Radu Tudor Ionescu, Andrei Butnaru, and Tommi Jauhiainen. 2019. A Report on the Third VarDial Evaluation Campaign. In *Proceedings of the Sixth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*. Association for Computational Linguistics.

Marcos Zampieri, Preslav Nakov, and Yves Scherrer. 2020. Natural language processing for similar languages, varieties, and dialects: A survey. *Natural Language Engineering*, 26(6):595–612.