Italian dialect Identification Progress Report (Vardial 2022 shared task)

Computational Semantics for NLP, Spring 2022

Quynh Anh Nguyen, Giacomo Camposampiero and **Francesco Di Stefano**
ETH Zurich - Swiss Federal Institute of Technology
{quynguyen, gcamposampie, fdistefano}@ethz.ch

1 Introduction

Dialect classification represents a key task in the improvement of many other downstream tasks such as opinion mining and machine translation, where the enrichment of text with geographical information can potentially result in improved performances for real-world applications (Zampieri et al., 2020).

As a result, the interest in the study of language variation has been steadily growing in the last few years, as highlighted by the increasing number of publications and events related to the topic. However, little has been done so far by researchers in the context of automatic dialect recognition for the Italian language.

In this context, the *Languages and Dialects of Italy* (ITDI) task of VarDial Evaluation Campaign 2022[1] aims to bridge this gap, facilitating the development of models capable of properly classifying 11 regional languages and dialects from both Italy's mainland and islands (Piedmontese, Venetian, Sicilian, Neapolitan, Emilian-Romagnol, Tarantino, Sardinian, Ligurian, Friulian, Ladin, Lombard).

The shared task organizers provide a dataset consisting of a large pool of Wikipedia articles written in one of these dialects, in the form of Wikipedia dump. The task is closed and, therefore, participants are not allowed to use external data to train their models (exception done for off-the-shelf pretrained language models from the HuggingFace model hub or similar, the use of which however has to be clearly stated). The predictions are evaluated at sentence level using F_1 score.

2 Related works

In this section, we will briefly introduce several methods (Zampieri et al., 2020) which used to be applied on classification problems on similar languages or different dialects of a language.

[1] https://sites.google.com/view/vardial-2022/shared-tasks

Italian dialect identification SoTa For what concerns Italian dialect identification, there are several researches working on analysing Italian dialects features (Zugarini et al., 2020) but no previous experiments deal with language identification task is found. Thus, our scope is to find a way to tackle this problem drawing inspiration from related works in dialect identification in different languages rather than overcome a particular state of the art model. In particular, we aim to obtain good results with the use of deep neural networks, more specifically adopting CNN and transformer architectures.

Machine learning models vs CNN Although deep learning models yield state of the art performances in many NLP tasks, an ensemble of SVM and Naive Bayes models was the best performing model in the Uralic Language Identification task in the VarDial Evaluation Campaign 2021 (Ceolin, 2021). *Linear SVM* classifier, *Naive Bayes* model, the combination of the two methods *Linear SVM + Naives Bayes* as well as *CNN* were implemented to classify target languages. Three machine learning approaches were all trained on TF-IDF character n-grams. The experiment shows that CNN did not always outperform the other machine learning approaches.

CNN Even if the state of the art in many dialect identification tasks has been reached through the application of transformer-based models, the use of CNNs is still high in this type of task. In particular, taking as an example the Romanian vs Moldavian dialect, CNN-based approaches achieved competitive results in both VarDial 2019 Evaluation Campaign (Tudoreanu, 2019) and VarDial 2020 Evaluation Campaign (Rebeja and Cristea, 2020).

Transformer The introduction of transformers (Vaswani et al., 2017) has revolutionized many tasks in NLP and the identification of dialects isn't

an exception. Models based on this architecture achieved state-of-the-art results in many application. A recent example is again VarDial 2020 Evaluation Campaign, where the use of a fine-tuned version of BERT previously trained on three publicly available Romanian corpora (Zaharia et al., 2020) reached a weighted F_1 score of 96.25% on the MOROCO dataset (Butnaru and Ionescu, 2019) in the Romanian vs Moldavian identification task.

3 Dataset

The training dataset is provided by the organizers and consists of 265 016 selected Wikipedia articles from 1st March 2022 dumps, comprehensive of all the 11 dialects evaluated in the task. The development set consists of 6800 annotated sentences that cover only 7 out of the 11 dialects evaluated in the shared tasks (there are no development samples for Emilian, Neapolitan, Ladin and Tarantino dialects). The test set has not been discloed yet, and will be made available to the participants according to the shared task schedule.

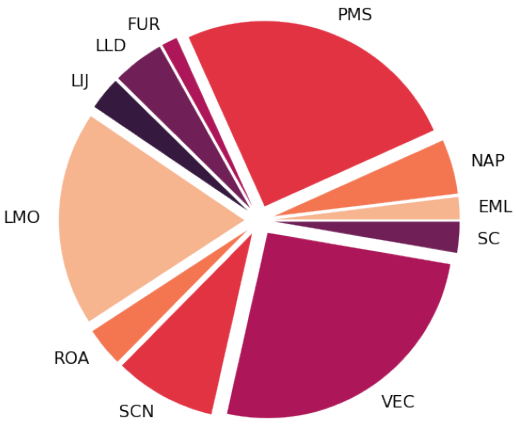
Since the data doesn't come from a well-know documented dataset, a preliminary exploration has been initially conducted to gain useful insight about it. This investigation highlighted a huge imbalance between classes as shown in Figure 1a, since the 3 most represented dialect (Venetian, Piedmontese and Lombard) account for almost three quarters of the entire articles in the training data. However, as shown in Figure 1b, the number of articles per dialect might not be fully representative on itself, since the size of each articles has also to be taken in account. This is the case of Lombard dialect, for example, which accounts for a large portion of training articles (19%), but more than a quarter of them are not longer than 2 sentences.

Nonetheless, imbalanced data seems to represent the main challenge presented by this dataset and should be addressed during the evaluation of the model. Possible solutions to the problem are data augmentation, weighted modelling and data subsampling.

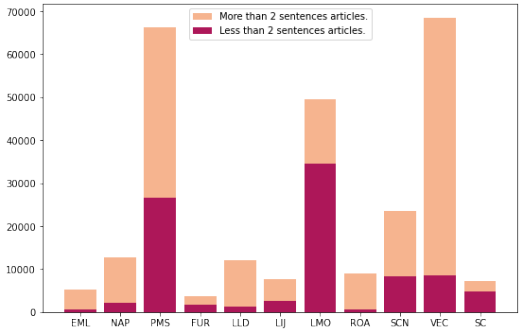
4 Methods

In order to tackle the Italian dialect identification tasks, we plan to investigate different approaches, including Machine Learning methods and Deep learning methods.

ML approaches Linear SVM and Naive Bayes are the two methods which are still popularly used



(a) Distribution of Wikipedia articles across dialects.



(b) Number of sentences of articles in the training set, grouped by dialect.

Figure 1: Preliminary data exploration on training set.

to handle this specific task. Thus, we would like to implement them with tf−idf character n-grams varying from 3 to 5 grams as in the experiment of (Ceolin, 2021). These approaches could be also used as baseline models to compare to deep neural network approaches such as CNN or Transformers.

CNN Convolutional Neural Networks were employed with success by many teams in the previous editions of the dialect identification task. Therefore, we plan to experiment with a classifier that uses a CNN for feature extraction on top of an embedding layer - that could be a skip-gram model (Mikolov et al., 2013) as in the case of (Tudoreanu, 2019).

Transformer The use of transformer-based models yield state of the art results even in the task of dialect identification. In particular, the fine-tuning of pre-trained BERT models obtained good results in this field. Following this line, our approach with this type of model would be to adopt a version of BERT pre-trained with the Italian language (Polignano et al., 2019) and to fine-tune it on our task.

5 Preliminary Results

5.1 Pre-processing of the Wikipedia dumps

The training data is provided in the form of raw Wikipedia dumps and, as highlighted by the organizers, a careful pre-processing is an important part of the task. An initial pre-processing is performed using WikiExtractor[2], a Python script that extracts and cleans text from Wikipedia database backup dumps. The use of this particular tool for extraction was suggested by the organizers of the shared task. However, a careful qualitative analysis of the resulting text samples pointed out the need for a more fine-grained pre-processing, that was implemented as follows.

Firstly, we remove all the HTML tags (e.g. `<br>`, `&amp;`, etc.) and Wikipedia meta information (e.g. contributors, timestamps and comments) that were not captured by WikiExtractor. Then, we observe that most of the documents of length < 50 are noisy observations, that come from documents for which WikiExtractor failed to extract any text at all or pages that contain simple and repetitive name entity definitions (e.g. small towns or years articles). Hence, we remove from the training dataset all such documents. Moreover, we observe that the training set contains duplicate documents (e.g. Web domains pages in Venetian Wikipedia). Therefore, we remove all the duplicates in the remaining dataset. Finally, since the task evaluate dialect classification at sentence level, we split all the documents into sentences using an Italian spaCy tokenizer. After the splitting, a further filtering is applied to the sentences, to trim a huge set of almost-identical sentences from the training data.

The exact number of documents after each pre-processing step is shown in Table 1, while a representation of the distribution of the input sentences over all the 11 dialects can be found in Figure 2.

Pre-processing steps	# documents
Original documents	265016
remove length < 50	245193
remove duplicates	219178
sentence split	562495
sentence cleaning	388781

Table 1: Number of training documents after each pre-processing step.

[2]https://github.com/attardi/wikiextractor

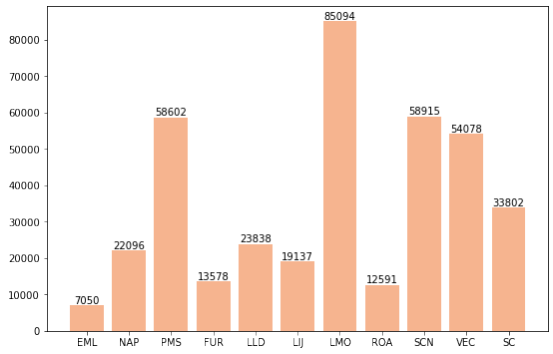


Figure 2: Number of sentences in the training set, grouped by dialect.

5.2 Baseline model

The first model we experimented with is Logistic Regression (LR), a simple model often used as baseline for classification tasks. The model is trained on scaled word-level TF-IDF features and evaluated on the development set of the shared task. The LR solver used for the experiment is `saga`, particularly indicated for large datasets. The results achieved by this baseline model, shown in Table 2, are surprisingly good, with a micro F_1 score of 0.93. Note that the dialects missing in the table (Emilian, Neapolitan, Ladin and Tarantin) are not included in the result table as no samples of these dialects are included in the development set.

Dialect	Precision	Recall	F_1	Support
PMS	0.98	0.98	0.98	1191
FUR	0.99	0.99	0.99	676
LIJ	0.97	0.99	0.98	617
LMO	0.80	0.96	0.87	1231
SCN	0.95	0.96	0.96	1371
VEC	0.97	0.79	0.87	1236
SC	0.97	0.82	0.89	477

Table 2: Baseline model evaluation.

We speculate that the great performances of this method depends on the huge linguistic variety between the evaluated dialects, that allows for a neat linear separation of the different classes in the feature space induced by TF-IDF.

This hypothesis seems to be partially corroborated by the development set confusion matrix in Figure 3. This matrix clearly shows how the most confused dialects are Venetian and Sardinian with Lombard and Sicilian respectively. The two pairs of dialects in fact share many cultural and geographical features, and might share similar words.

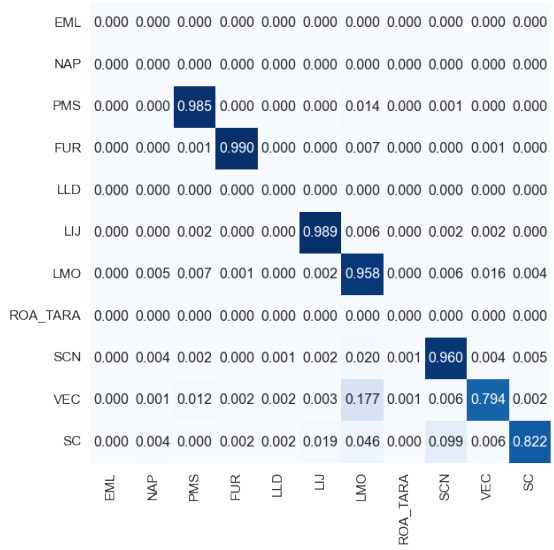


Figure 3: Baseline approach confusion matrix on the development set.

5.3 CNN

CNN has been implemented in both character level and word level. Accuracy and F_1-macro are two evaluation metrics that used to evaluate models' performances. The details of implemented models and model loss training over different setting of epochs number are provided in the Appendix section with the table **??** and Figure 5.

By *manually* implementing different sets of hyper-parameter, we aim to find the better model architecture and training regime for classifier tasks. Several hyper-parameters, including *optimizer, loss function, learning rate, dropout, strides sizes, kernel sizes, batch sizes, embedding size*, are taken into consideration in our experiment.

Table 3 shows the classification results of two CNN models over different number of epochs. The best performance is achieved from CNN model tokenized in character level training over 20 epochs. In general, there is no significant improvement when training and testing data with CNN char-level and CNN word-level. On the other hand, running time cost on CNN word-model is much more expensive compared to the same setting running on CNN char-level. The different regarding vocabulary size of CNN word-level models and CNN character-level models might be the main reason that leads to the different in term of computational cost. In fact, the vocabulary size of CNN word-level models and CNN character-level models shown in the table 3 are respectively 989 and 788, 197 tokens.

The best CNN model achieves approximately 86% accuracy and 55% F_1-macro score on the validation data. It shows a big gap in performance compared to other traditional machine learning approaches that mentioned in Section 5.2.

There are two main reasons why Convolutional Neural Network could not perform better than other ML classifiers. First, bad result of CNN might be the consequences of how text are embedded. In fact, we encoded text in character-level/word-level with different embedding sizes. However, single character, i.e. 1-gram, is the only way encoded the text. Meanwhile, by using scikit-learn for classifiers models, we encoded texts with different configurations including word levels, character level and character within boundary of word level. Second, CNN might be more complicated than classifier methods to handle our dataset.

In general, a powerful model has tendency to treat simple problem with complicated architecture. This leads to over-fitting issue which indicates that our model is too complex for the problem that it is solving. As consequence, the model resulting from CNN performs badly on the unseen data.

Tokenization level		**Char-level**			**Word-level**	
Epochs		5	10	20	5	10
Training set	**Accuracy Score**	0.942317	0.95291	0.96428	0.93953	0.95610
	F_1-macro Score	0.93468	0.94718	0.95999	0.93144	0.94973
Validation set	**Accuracy Score**	0.84212	0.85554	**0.86055**	0.82989	0.85126
	F_1-macro Score	0.54154	0.54808	**0.55083**	0.53127	0.54594

Table 3: CNN models evaluation in character-level and word-level.

5.4 Transformer

Moving towards state-of-the-art models, we experimented with a well-known pre-trained language model: BERT (Devlin et al., 2019). More specifically, we fine-tuned on the ITDI task three different BERT models from HuggingFace:

- $\text{BERT}_{\text{LARGE}}$ pre-trained on Italian corpora
- $\text{BERT}_{\text{BASE}}$ pre-trained on Italian corpora
- multilingual $\text{BERT}_{\text{BASE}}$ pre-trained on corpora comprehensive of 102 languages.

We observed that the BERT model that achieves the best performance on the development set is the first, i.e. $\text{BERT}_{\text{LARGE}}$ pre-trained on Italian corpora. As we can observe from the normalised confusion matrix in Figure 4, the performances of this pre-trained BERT resemble a lot the results obtained by the baseline model. It is interesting to observe that also in the Transformer-based model predictions the most confused dialects are respectively Venetian with Lombard and Sardinian with Sicilian. These results support the hypothesis of an intrinsic difficulty in the discrimination between these pairs of dialects, related to linguistic features of the dialects rather than model-dependent.

Even if the performances are similar to the baseline ones in the classification of most of the dialects, the $\text{BERT}_{\text{LARGE}}$ model obtain worse results for the two more difficult dialects to classify, Venetian and Sardinian. Overall, this results in an F_1-micro score of 0.88 (0.05 less than the result achieved by the baseline model). In Table 4, we can see the accuracies and F_1-micro scores obtained by the three pre-trained BERT models.

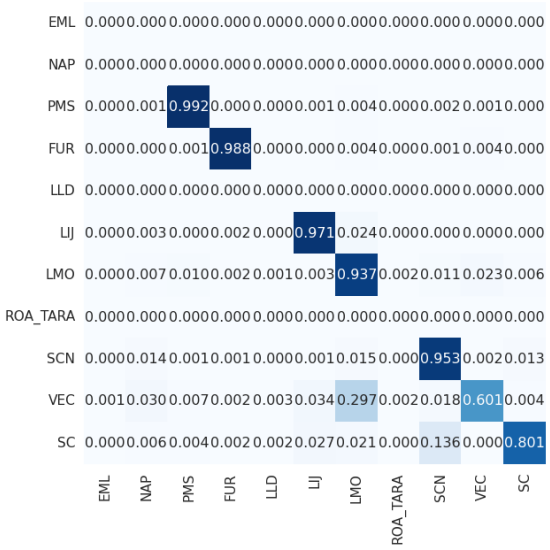


Figure 4: Confusion matrix on the development set of the pre-trained BERT large on the Italian language.

6 Future experiments

Considered the competitive results obtained by the baseline model, the exploration of other classical machine learning models for classification (as well as ensembles of these models) represents a promising route of improvement. Other embedding techniques, e.g. Hashing Vectorizer, could be investigated as well.

Regarding CNN model, using automatic libraries such as RayTune to tune hyper-parameters might improve the CNN model performance. As result shown in section 5.3, the trade-off between running time and the improvement of model between CNN char-level and CNN word-level is not significant. Thus, we plan to exclusively optimize CNN in character tokenization level.

Moreover, in order to handle the class imbalances issue, data augmentation could be take into consideration.

For what concerns the state-of-the-art approach, a possible improvement could rely on the use of an ensemble of the different pre-trained BERT models if these make wrong predictions on different samples from the development set.

Finally, a careful analysis of the factors that lead to the mis-classification of Venetian and Sardinian samples (with Lombdard and Sicilian respectively) should be tackled as well, as it might result in a considerable improvement of performances in the classification of these two dialects.

BERT model		Italian base	Italian large	multi-lingual base
Validation set	**Accuracy Score**	0.87692	**0.88722**	0.87119
	F_1-micro Score	0.87692	**0.88722**	0.87119

Table 4: BERT models evaluation.

References

Andrei M. Butnaru and Radu Tudor Ionescu. 2019. Moroco: The moldavian and romanian dialectal corpus.

Andrea Ceolin. 2021. Comparing the performance of CNNs and shallow models for language identification. In *Proceedings of the Eighth Workshop on NLP for Similar Languages, Varieties and Dialects*, pages 102–112, Kiyv, Ukraine. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, page 3111–3119, Red Hook, NY, USA. Curran Associates Inc.

Marco Polignano, Pierpaolo Basile, Marco de Gemmis, Giovanni Semeraro, and Valerio Basile. 2019. AlBERTo: Italian BERT Language Understanding Model for NLP Challenging Tasks Based on Tweets. In *Proceedings of the Sixth Italian Conference on Computational Linguistics (CLiC-it 2019)*, volume 2481. CEUR.

Petru Rebeja and Dan Cristea. 2020. A dual-encoding system for dialect classification. In *VARDIAL*.

Diana Tudoreanu. 2019. DTeam @ VarDial 2019: Ensemble based on skip-gram and triplet loss neural networks for Moldavian vs. Romanian cross-dialect topic identification. In *Proceedings of the Sixth Workshop on NLP for Similar Languages, Varieties and Dialects*, pages 202–208, Ann Arbor, Michigan. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

George-Eduard Zaharia, Andrei-Marius Avram, Dumitru-Clementin Cercel, and Traian Rebedea. 2020. Exploring the power of Romanian BERT for dialect identification. In *Proceedings of the 7th Workshop on NLP for Similar Languages, Varieties and Dialects*, pages 232–241, Barcelona, Spain (Online). International Committee on Computational Linguistics (ICCL).

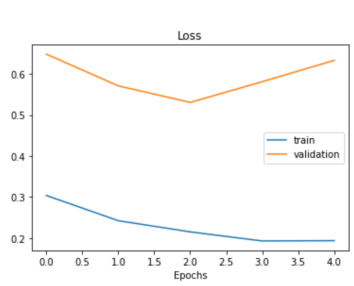
Marcos Zampieri, Preslav Nakov, and Yves Scherrer. 2020. Natural language processing for similar languages, varieties, and dialects: A survey. *Natural Language Engineering*, 26(6):595–612.

Andrea Zugarini, Matteo Tiezzi, and Marco Maggini. 2020. Vulgaris: Analysis of a corpus for middle-age varieties of italian language. *CoRR*, abs/2010.05993.

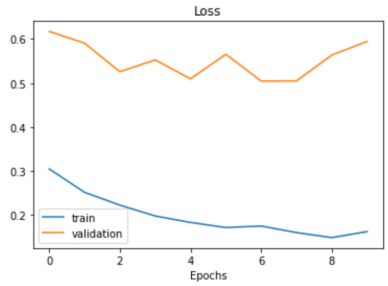
Appendix A: CNN

Tokenization	CNN Model Summary
Character level	(embeddings): Embedding(989, 512) (conv2d): Conv2d(1, 16, kernel_size=(3, 3), stride=(2, 1), padding=(1, 0)) (max_pool2d): MaxPool2d(kernel_size=(6, 12), stride=(2, 1), padding=(1, 0), dilation=1, ceil_mode=False) (conv2d_2): Conv2d(16, 16, kernel_size=(6, 6), stride=(2, 1), padding=(1, 0)) (max_pool2d_2): MaxPool2d(kernel_size=(6, 12), stride=(2, 1), padding=(1, 0), dilation=1, ceil_mode=False) (linear): Linear(in_features=7728, out_features=12, bias=True)
Word level	(embeddings): Embedding(788197, 512) (conv2d): Conv2d(1, 16, kernel_size=(3, 3), stride=(2, 1), padding=(1, 0)) (max_pool2d): MaxPool2d(kernel_size=(6, 12), stride=(2, 1), padding=(1, 0), dilation=1, ceil_mode=False) (conv2d_2): Conv2d(16, 16, kernel_size=(6, 6), stride=(2, 1), padding=(1, 0)) (max_pool2d_2): MaxPool2d(kernel_size=(6, 12), stride=(2, 1), padding=(1, 0), dilation=1, ceil_mode=False) (linear): Linear(in_features=7728, out_features=12, bias=True)

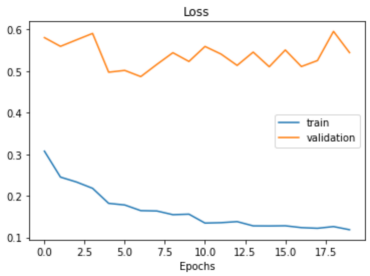
Table 5: CNN Model Summary



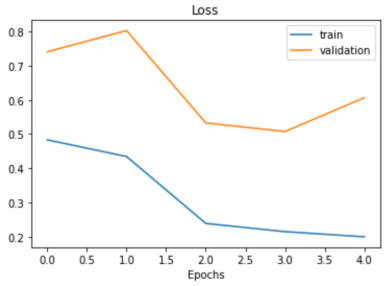
(a) Char-level model loss over 5 epochs



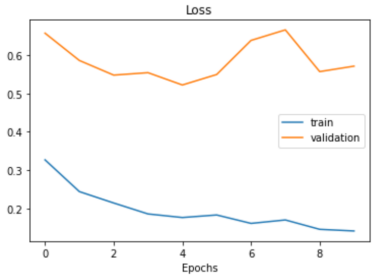
(b) Char-level model loss over 10 epochs



(c) Char-level model loss over 20 epochs



(d) Word-level model over 5 epochs



(e) Word-level model over 10 epochs

Figure 5: CNN models loss vs epochs.