Next-Word and Variant Prediction Algorithm for Sardinian Dialects

Evan Chapple

5/6/23

Abstract

In this paper, we present a next-word prediction algorithm for the four main variants of the Sardinian language: Campidanese (Camp), Logudorese (Logu), Nugorese (Nugo), and Comuna (Comu). We collected a corpus of Sardinian texts from a Wikipedia dump, which was then processed to add variant tagging. We then developed a neural language model using a Long Short-Term Memory (LSTM) architecture to predict the next word in a sentence given the context of the previous words and the variant of the sentence. We evaluated the model's performance by comparing it to a single-task model using clicks saved as a metric and found it performed only slightly worse. Our approach demonstrates that even for languages with limited data resources, computational models can still be used to provide valuable assistance in language production.

1 Introduction

With the development of handheld electronic devices over the past 100 years, there has been a revolution in the way in which people interact with language. Writing went from being painstakingly scratched onto parchment to generative models that create mountains of text from a short prompt. While the latter has been shown to be incredibly effective for languages that have a large online presence (Zhang and Li, 2021), there are many languages that do not have such an abundance of data that can be used as input to train such models. There has, however, been a lot of work on the creation of computational models that do not require such large corpora to be trained .

In this thesis, a model that serves to jointly predict the next word and variant of a particular stream of characters, specifically for the different Sardinian dialects, will be presented. This model is based on a next-word prediction algorithm that is also able to recognize which variant is being used. The model is trained on the character level. To evaluate the effectiveness of our model, we conducted validation tests on a subset of our Sardinian dataset, which will be discussed later. The purpose of this effort is to create a model that is able to adapt word predictions to the user's variant

In this paper, we will begin by discussing the variants of interest and their usage within the island of Sardinia. Afterwards, we will describe the process of collecting and cleaning our corpus. Then, we will look at works related to the work presented in this project followed by our methods. The results of our model will then be compared with a previous, less advanced model and these results will be discussed.

1.1 Sardinian Variants

Sardinian, as a Romance language, shares many linguistic features with other Romance languages, especially those that it is geographically near to, such as Italian, French, Catalan, and Spanish, but also retains some unique characteristics due to its isolation on the island (Wheeler, 2005). There are certain characteristics of modern Sardinian that lead some scholars to believe it is closer to Latin than other romance languages due to its relative isolation (Tufi, 2013). The phonology, morphology, and syntax of Sardinian dialects present distinct features that differentiate them from other Romance languages, as well as from one another (SAR, 2006).

The variation found among Sardinian dialects can be attributed to the island's geographic isolation, historical events, and the influence of various populations that have occupied Sardinia over the centuries, such as the Phoenicians, Carthaginians, Romans, Vandals, and Byzantines (Dyson and Rowland, 2007; D., 2019). These factors have contributed to the development of a rich linguistic landscape, making the study of Sardinian dialects a fascinating area of research for linguists.

Despite the differences among dialects, speakers from different regions of Sardinia can generally understand each other, although some communication difficulties may arise due to differences in vocabulary and pronunciation (D., 2019). The future of Sardinian is still in danger however, as speakers who learn Sardinian as a child do not seem to maintain the language into adulthood as Italian becomes their main language (Moseley, 2010). This has led Sardinian to still be listed as an endangered language despite its relatively large speaker base (Lewis, 2009). In Figure 1, one can see the geographical regions where the various dialects are spoken. In an effort to promote the use of Sardinian and to facilitate communication among speakers

Variant	Text
Comu	*babbu nostru chi ses in is chelos*
Camp	*babbu nostu ki ses in is celus*
Logu	*babbu nostru k'istas in sos kelos*
Nugo	*babbu nostru, ch'istas in sos chelos*

Table 1: Reconstructions of the beginning of the Lord's prayer in each of the dialects treated in this paper.

of various dialects, the introduction of the Comu orthography aimed to provide a common written form for the language. Comu is a standardized orthography created by the Autonomous Region of Sardinia for use in official documentation along with Italian beginning 2006 (Mooney, 2017). However, it has faced resistance from some speakers who prefer to use their local dialects' orthography or who find the standardized form too restrictive.

Efforts to preserve and promote the Sardinian language have increased in recent years. Educational initiatives, such as teaching Sardinian in schools alongside Italian, have been introduced in an attempt to ensure its survival (Lupinu et al., 2007). Despite these attempts, the number of speakers continues to decrease, and Sardinian is only heard sporadically in everyday life (D., 2019).

In Table 1, one can see a set of equivalent sentences, the beginning of the Lord's Prayer, in each of the four variants. It is important to note that while some of these sentences differ somewhat that the orthographies of all variants apart from Comu are not standard and can differ from writer to writer. For example, the word *chi* is commonly replaced by *ki*, an older spelling that is otherwise equivalent SAR (2006). Comu, itself, is not completely standard, and allows for flexibility, for instance, *is* and *sos* are interchangeable and acceptable forms of the masculine plural article.

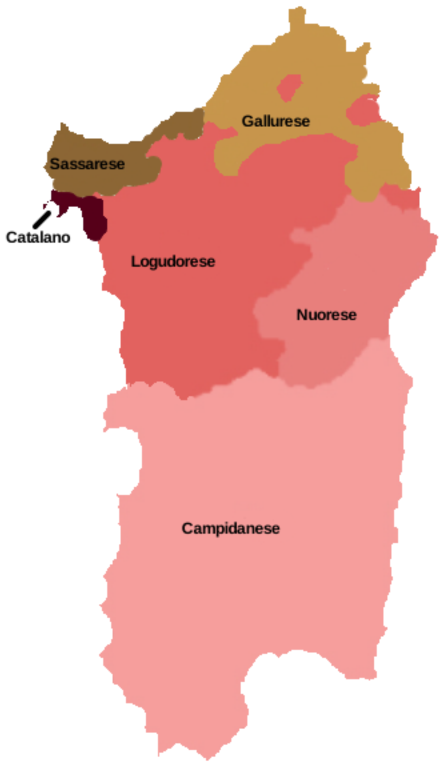

Figure 1: A map showing the main dialectal regions of Sardinia (Ciccolella, 2015)

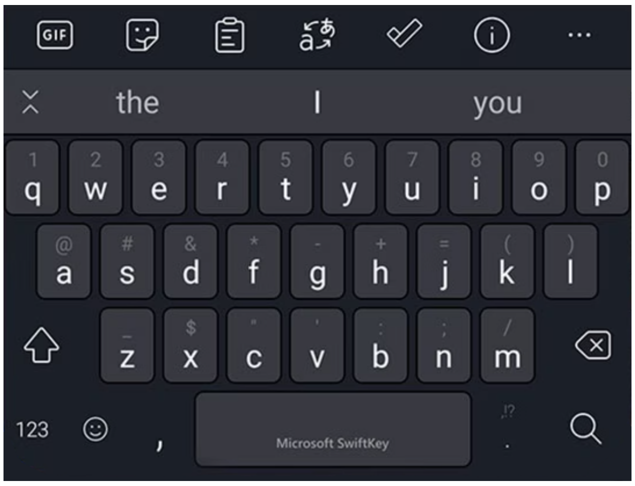

Figure 2: An image of Microsoft Swiftkey's keyboard demonstrating its 3 predictions (Microsoft, 2023)

2 Related work

2.1 Predictive Keyboards

Predictive keyboards have become a mainstay for many cell-phone users who speak a majority language (Microsoft, 2023). A modern example of this can be seen in Figure 2. This technology lets us interact with our device in a more efficient manner. Many approaches to tackle the creation of predictive keyboards use probabilistic measures based on how likely the next character is likely to be inputted based on the previous combined with distance from the key-input in order to perform word prediction or correction (Vertanen et al., 2015; Zhai et al., 2002).

2.2 Language Modeling

Research in the area of predictive keyboards and language models for low-resource languages has progressed significantly in recent years . The application of techniques like transfer learning and cross-lingual resources has shown positive results in improving the performance of such models for these languages (Tran et al., 2018; Kim et al., 2016).

Much recent research has found that RNNs and LSTMs are effective for various natural language processing tasks, including next-character prediction (Chung et al., 2014; Graves, 2013). Gated Recurrent Units (GRUs) have also been proposed as an alternative to LSTMs that are also able to make long-term dependencies, discussed later, while being less computationally demanding (Cho et al., 2014). This is important to note as mobile devices still do not do not all have the ability to run costly models.

In addition to these techniques, researchers have proposed methods for creating multi-task learning systems that can predict both characters and other linguistic properties, such as language variant or part-of-speech tags, which can further improve model performance (Ruder, 2017).

Another improvement made to sequence based machine learning tasks is beam search. Beam search is a widely-used search strategy for sequence prediction tasks that often outperforms greedy algorithms by expanding over n best predictions at the same time (Ranzato et al., 2016; Huang et al., 2018). After every step, the n most probable partial sequences, also known as beams, are expanded upon while the rest are discarded. This approach allows for the benefits afforded by an exhaustive search, breadth-first or depth-first, while still remaining relatively computationally efficient.

The primary advantage of beam search is its ability to balance the search space exploration and computational efficiency. Using a fixed beam width representing the number of hypotheses considered at each step, the algorithm can avoid getting stuck with locally biased results and instead explore other, more promising beams. Beam search has been successfully applied to a variety of sequence prediction tasks, such as machine translation, speech recognition, and more, where its ability to consider wider contexts allows it to shine (Sutskever et al., 2014; Chorowski et al., 2015).

Certain metrics have been proposed as a baseline measure with which to compare the performance of different keyboards, such as word error rate, clicks saved, and Keystroke Savings Rate (KSR) (Vertanen and Kristensson, 2011; Boudreau et al., 2020). The formula for calculating KSR is

$$\text{KSR} = (Keys_{normal} - Keys_{prediction})/Keys_{normal} * 100 \tag{1}$$

For the purposes of this paper, as we focus on next-word prediction, we will focus on clicks saved and KSR. Such metrics provide valuable insight into the performance of the predictive keyboard.

Finally, the process of collecting and preprocessing corpora for low-resource languages for modeling is not a straightforward task. There has been much research exploring the use of techniques such as web scraping to easily collect large amounts of text (Alonso and Plank, 2020). Furthermore, text normalization acts as an important step when processing the varied and sometimes messy text that would be collected. Furthermore, there are also techniques that have been proposed to combat the effects of having unequal datasets, such as undersampling/oversampling and data augmentation (Mohammed et al., 2020; Ayana and Menzel, 2017). An example of oversampling, which we use in our corpus, is seen in Figure 3.

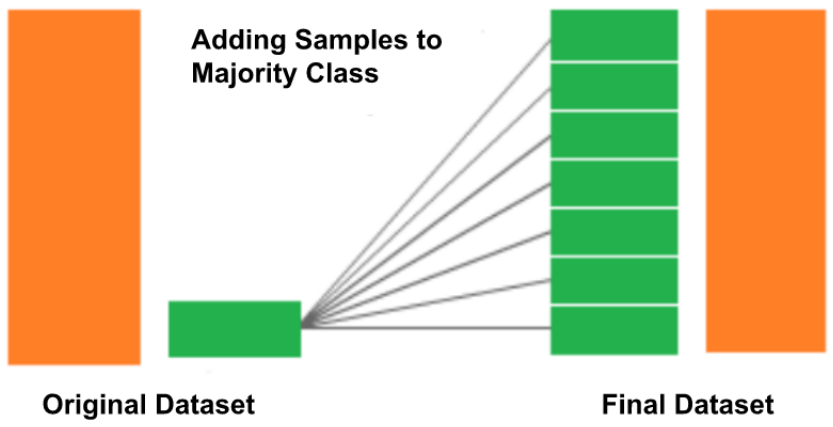

Figure 3: A demonstration of how two relatively equal datasets are created from originally unequal datasets via oversampling (Mohammed et al., 2020)

2.3 NLP for Language Variants

Recent research in natural language processing has increasingly focused on the challenges presented by language variants, particularly for low-resource languages and dialects, as creating useful resources for these at-risk languages is the only way to preserve them for generations to come. Several studies have addressed the issues of language identification for these language variants (Ali et al., 2017; Jaech et al., 2016).

There have been language variant identification systems proposed that distinguish between several closely related dialects of the Arabic language (Ali et al., 2017). This model utilized character and word n-grams to build a classifier that achieved high accuracy in identifying these dialects. There was another project that worked on language identification in short code-switched texts, collected from tweets (Jaech et al., 2016). This type of text is particularly difficult as it includes instances where multiple languages or dialects are used within a single sentence. This mode also introduced a novel approach for language identification in code-switched text, leveraging both first character level vector representations, using char2vec, then word level n-grams for language identification.

The VarDial Evaluation Campaign has played a significant role in promoting research on language variants, particularly for low-resource languages and dialects. They do so by proposing tasks that research groups from around the world participate in. VarDial 2022's tasks required participants to develop and implement a model that would identify/classify Italian languages and dialects, such as Sicilian, Lombard, and Piedmontese, and Sardinian among others (Aepli et al., 2022; Tommi Jauhiainen and Lindén, 2022). By targeting these specific languages, VarDial 2022 aimed to further advance the state of the art in NLP research for these language variants, while also promoting awareness of the rich linguistic heritage of Italy.

3 Methods

The goal of this project was to develop a predictive keyboard for Sardinian speakers by creating a model that could predict the next character and the corresponding variant given an input stream of characters. Such technology would help improve speed and accuracy of Sardinian speakers and facilitate communication in the language.

3.1 Corpus

The corpus used to train the next-word prediction algorithm was collected from a Wikipedia dump on September 20, 2022. The dump contained texts in Sardinian (SC) in one of four main variants: Campidanese (Camp), Logudorese (Logu), Nugorese (Nogu), and Comuna (Comu).

To prepare the corpus for training the model, the texts were extracted from the Wikidump and tagged with a relevant variant label. All non-content text was removed including the removal of any Wiki Markup, otherwise known as Wikitext. This preprocessing step ensures that the training data contains only the raw text without any formatting or structural elements.

Furthermore, the texts were split on '.', ',' ':' and ';', except for in the case of numbers which were often formatted as '9.200'. Next, all non-alphanumeric characters were removed from the corpus apart from '''. The hope was that the removal of these elements would allow the model to not worry about predicting such characters.

Any characters that appeared fewer than 40 times were inspected and eventually replaced with more standard forms (e.g., *ý* (1 instance) → *y*). This character-level normalization would help the model generalize better and since there were so few instances of these characters to begin with, the model would most likely not have predicted them anyways.

Finally, in order to ensure the consistency of the corpus, a manual inspection was conducted. All examples of non-variant language were removed, and other corrections were made as needed. This step was crucial for maintaining the quality and representativeness of the training data, as it helps eliminate potential entries for inaccuracies that would not be covered by the former strategies.

In order to account for the uneven distribution of text across variants and to mitigate bias caused by unequal distribution of input, variants that had significantly fewer sentences were copied until they reached approximately 1,000 sentences in length. This process, described earlier as oversampling, served to balance the dataset and ensure that each variant would have an equal opportunity to contribute to the model's learning. See Table 2 for information about this.

Variant	Code	Multiplier	Sentences	Tokens
Limba Sarda Comuna	sc-comu	1	1,002	105,355
Campidanesu	sc-camp	1	765	84,710
Logudoresu	sc-logu	2	493	61,826
Nugoresu	sc-nugo	26	38	3,107

Table 2: Statistics on number of sentences and tokens for each of the variants in the corpus. The multiplier is the amount of times the sentences were oversampled in order to reach a balanced corpus of 1,000 per variant.

With the aforementioned modifications, it is hoped that the corpus provides a representative sample of the Sardinian language. The corpus was curated and balanced in order to train and evaluate the next-word prediction algorithm presented in this thesis. The careful preprocessing of the corpus ensures that the resulting model is accurate and representative, and should be able to distinctiveness of each variant.

3.2 Model

The prediction model was implemented using PyTorch, a machine learning library for Python (Paszke et al., 2019). Using this library, we decided to implement a recurrent neural network (RNN) model as such models have been shown to perform well in pattern recognition tasks in sequential data, such as character streams (R. et al., 2018). RNNs specifically perform better than Feed-Forward Networks, which employ an n-gram approach, due to its ability to perform context dependencies of unfixed length (Sundermeyer et al., 2013).

Now, let's see why RNNs perform the way they do. RNNs are a subset of artificial neural networks (ANNs) and are designed for processing sequential data. The advantage that RNNs are known for is in cases in which the input data is dependent on itself by having neurons attached to one another. At each step in the input, an

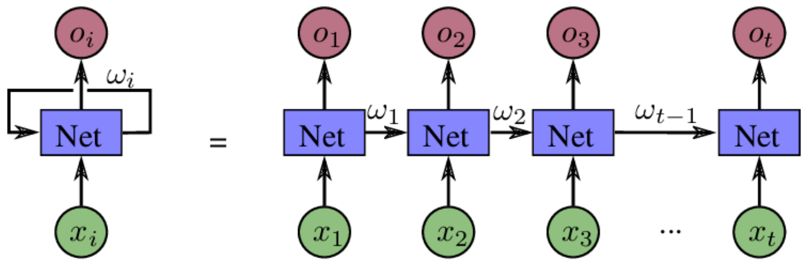

Figure 4: A folded (left) and unfolded (right) example of the architecture of an RNN model. x_i is the input to the model at point i, with o_i being the output, the hidden state w_i can also be seen being passed to the next step (Tai and Liu, 2016)

RNN processes the current input along with the previous step's hidden state, an output that represents the "state" resulting from the previous inputs and that is produced alongside the target output. These two pieces of information are then processed by the model to produce a new output and hidden state, given what data has come before. Two key advantages of RNNs are then that they are able to process sequences of varying lengths by changing the number of steps used to process an input and through using a process called backpropagation to capture long-term dependencies and patterns. An example of this RNN architecture can be found in Figure 4.

Furthermore, we chose to use a long short-term memory (LSTM) architecture because it is capable of capturing distant relationships in sequential data and addresses gradient vanishing, one major limitation within RNNs (Sundermeyer et al., 2012; Noh, 2021). The reason it can do so is because an LSTM is trained to forget or retain previous information in the input and update the current state accordingly. All together, such a model allows for more accurate predictions of the next character in a sequence than other models could offer.

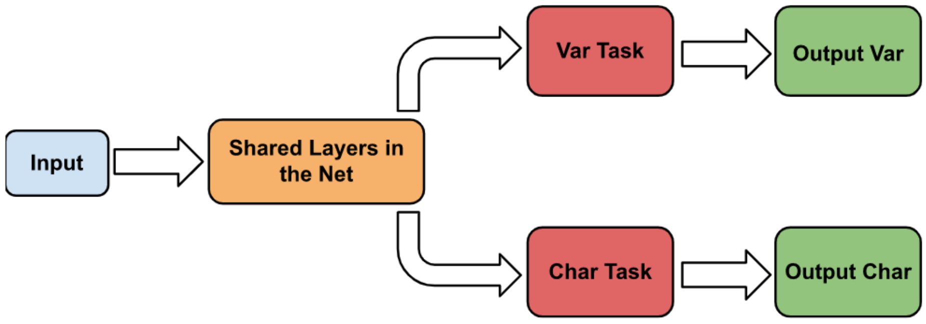

Figure 5: Multi-task architecture where the tasks are variant prediction and character prediction

To create multiple predictions, we built a model with a multi-task learning approach that enabled it to predict both characters and variants simultaneously. This approach eliminated the need to train two separate models and allowed us to use a single model that could produce both outputs. One can see an example architecture of such a model in Figure 5. It has been shown that multi-task models perform well on related tasks and can form relations between the two outputs to improve both outputs (Caruana, 1997; Liu et al., 2016; Ruder, 2017). To train the model in this way, we included the correct variant information along with the prime characters in the training data batches. We used a cross-entropy loss function to train the model and applied gradient descent to update the weights in the model.

Before implementing the multi-task system, we trained a single-task RNN model to predict only characters. The model itself was based on work done by (Barhate, 2019). The single-task model used a similar LSTM architecture as the multi-task model but with a different output layer. However, the single-task model did not perform significantly better than the multi-task system, as it predicted roughly the same number of characters in the test set. Therefore, we decided that the the multi-task model was still sufficiently effective and more convenient

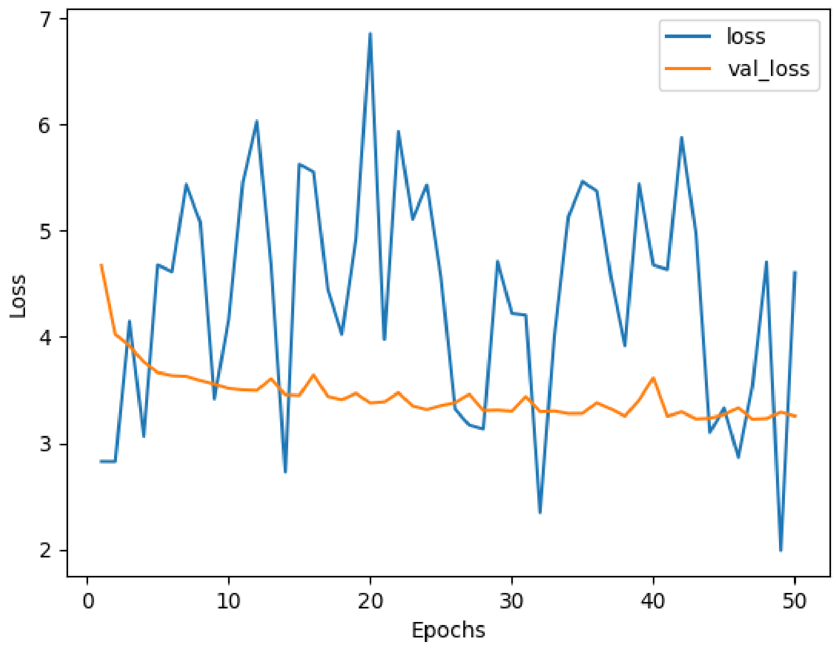

Figure 6: A graph mapping the validation loss (val_loss) and training loss (loss) of the model over the course of the training, measured in epochs

The data was split 80/10/10 for train/validation/test sets. We then evaluated the model's performance using two metrics: number of clicks saved and percentage of correct variant prediction. Clicks saved measures the number of clicks saved by a user using the prediction algorithm compared to a user that is using a regular keyboard. It is important to note that a correct prediction of length 1 results in 0 clicks saved and in the same vein a correct prediction of length 4 would result in 3 clicks saved. Furthermore, variant prediction accuracy measures the percentage of input sentences where the model predicts the correct, corresponding variant.

3.3 Training the Model

In the training phase of the model, various hyperparameters were tuned to optimize performance on both the prediction of the next character in the stream and of the current variant. Some of the key hyperparameters in this model include; the learning rate (0.001) which controls how quickly the model updates its weights, batch size (4) which is the amount of data passed in a single batch, drop probability (0.5) which is the likelihood that a neuron will be turned off during training, and number of epochs (100) which is the amount of times all the training data will be processed by the model. The best set of hyperparameters was chosen based on the lowest validation loss, which serves as a measure of the model's prediction error on a portion of the corpus that is not trained on. Figure 6 illustrates the validation loss curve, showing how the model's performance improved over the course of every epoch.

Early stopping was employed to prevent overfitting, which occurs when the model becomes too specialized on the training data and performs poorly on unseen examples. By monitoring the validation loss during training, the training process is halted when the validation loss starts to increase or plateaus, indicating that the model is no longer creating new, useful connections. After the training was completed, the final model was evaluated on a separate test dataset to assess its ability to generalize to new, unseen data, demonstrating the effectiveness of the chosen hyperparameters and the overall robustness of the model.

3.4 Evaluating the Model

After training, the model must be evaluated on novel data, the 10% of the corpus set aside for testing earlier. The test data, already split into a list of sentences along with the corresponding variant tag, is passed to the model a single character at a time. At which point, the model creates n predictions, character by character, until a white space character is reached. For the purposes of our testing, we had $n = 3$. These n predictions are then compared to the correct word ending, and the amount of correctly predicted characters are added to a tally and the length of the prediction is also saved. If the model predicts the correct sequence to finish the word, then the next step of the prediction begins at the beginning of the next word. This prevents padding the clicks saved statistic with prediction sequences that would have already be predicted. One important note is that when measuring clicks saved, the space directly following a word is not counted in the prediction length. The results of this model were then predicted to those of the single-task model.

To accurately measure the performance of the variant prediction, variant predictions were only saved at the end of a sequence. This does give an advantage to the model as it does not have to deal with variant recognition of short or unfinished sequences.

4 Results

All our results were collected as described in the methods section. As seen in Table 6, we can see that the single-task model performed slightly better than the multi-task model, statistically better (p<.0001) using a two-proportion z-test. These results suggest that with the addition of the variant prediction, our model does lose out slightly in its performance on the character prediction task, but does entirely gain the ability to perform variant prediction.

On the topic of variant prediction, in Table 7, one can see the multi-task's per-

Frequency	Predicted Variant	Actual Variant
42	Logu	Comu
26	Comu	Logu
24	Camp	Comu
21	Camp	Logu
15	Logu	Nugo
11	Logu	Camp
11	Comu	Camp
9	Camp	Nugo
7	Comu	Nugo
1	Nugo	Comu
1	Nugo	Camp

Table 3: Character prediction statistics made by the multi-task model.

formance on variant prediction. In fact, this model's prediction is significantly better than random chance, measured by a one-proportion z-test ($p<.0001$). However, the prediction is still quite poor compared to what it could be, being at less than half. Some issues with the distance between the chosen dialects will be discussed in the Discussion section. Another area of interest related to variant prediction is whether there were patterns in which variants were confused for which other variants most often. We see in Table 3 that Comu was most often misidentified as Logu followed by the inverse. Furthermore, Nugo was predicted incorrectly the least number of times with one instance where the actual variant was Comu and one where the actual variant was Camp.

If one were to look at Figure 7, one can see that over the course of training, there were significant and constant improvements in the KSR and, while the variant prediction did not improve as consistently, it still trended positively. These data were collected on the test set. Furthermore, when the model was trained more than 50 epochs, neither the KSR nor the variant prediction percentage went up by a mean-

Chars Predicted (Clicks Saved)	Number of Occurrences
1 (0)	2,362
2 (1)	867
3 (2)	267
4 (3)	75
5 (4)	15
6 (5)	7
7 (6)	1
Total Clicks Saved	5.321
Total Chars	34,861

Table 4: Character prediction statistics made by the multi-task model.

Chars Predicted	Clicks Saved	Number of Occurrences
1	0	2,791
2	1	947
3	2	283
4	3	73
5	4	23
6	5	9
7	6	1
Total Clicks Saved		5,973
Total Chars	34,861	

Table 5: Character prediction statistics made by the single-task model.

ingful amount, and validation loss began to stagnate.

5 Discussion

5.1 Interpretation of Results

Our multi-task model did not perform as well as the single-task model, which suggests that there might be room for improvement in the multi-task learning approach. The results are still promising, however, and contribute to the development of natural

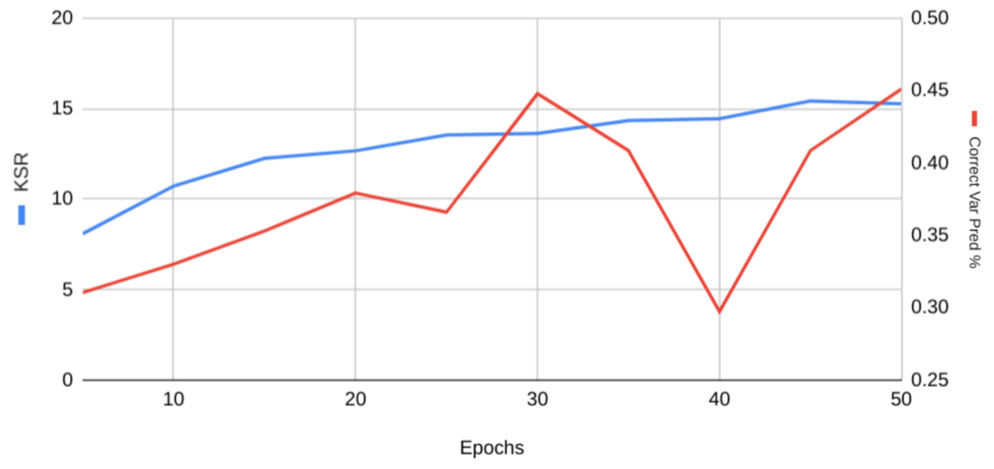

Figure 7: A graph mapping the KSR and the correct variant prediction algorithm over the course of training.

Model	**CpC**	**KSR**
No Prediction	1	0
Single Task	0.829	17.13
Multi-task	0.847	15.26

Table 6: Metrics used for measuring the peformance of a predictive keyboard. Clicks per Character (CpC) and Keystroke Savings Rate (KSR).

CVP	**TA**	**% CVP**
138	306	45.10%

Table 7: Variant prediction statistics made by the model showing correct variant predictions (CVP) and total attempts (TA)

language processing tools for the Sardinian language and demonstrate the potential of computational models to assist in the writing of low-resource languages. The findings have implications for the development of language technologies for other under-resourced languages and open up new opportunities for the use of these tools in various applications.

Looking at the variant prediction side of our multi-task model, one reason the performance could be poor is because the variants we chose to address are not discrete categories. According to authors including D. (2019), Logudorese and Nugorese are simply a singular dialect rather than two. If we consider them to be the same category, than our model's performance reaches just above 50%. However, as we saw earlier, the most common misidentifications were between Logu and Comu. This does make sense, as Comu was created taking parts of both Logu and Camp for inspiration SAR (2006). This also explains why the top 3 mispredictions all include Comu.

5.2 Limitations

It is important to realize that our study has a number of limitations. Firstly, the corpus used for training and evaluating the model was sourced from a Wikipedia dump, which may not be fully representative of the Sardinian language as a whole. Wikipedia articles tend to have a more formal register and may not capture the style and usage found in everyday communication.

Another limitation lies in the oversampling technique used to balance the dataset. While it helps ensure that each variant contributes equally to the model's learning, it may also introduce an artificial bias by duplicating sentences. This could potentially result in the model overfitting to certain patterns present in the oversampled sentences, limiting its ability to generalize to unseen data and possibly explaining the poor performance on predicting Nugo correctly as a variant. Alternative methods, such as data augmentation techniques or transfer learning from related languages, could be explored to mitigate this issue.

Lastly, our multi-task learning approach, while promising in theory, did not perform as well as the single-task model. This may suggest that the joint learning of character prediction task and variant prediction task could be suboptimal, or that the model architecture and training procedure was not properly refined for such a process. Exploring alternative multi-task learning strategies or employing task-specific model components might lead to improved performance in both tasks.

Addressing these limitations and considering potential improvements could further advance the development of predictive keyboards for Sardinian speakers and other explore techniques that could help other under-resourced languages.

5.3 Future Work

We will now mention some ways to address the aforementioned limitation of our process and other directions for future work on this topic.

The easiest way to improve this model is to collect more samples of each of the dialects. The severe lack of data for the Nugorese dialect makes it almost impossible for the model to have any real training on it, even with the over-sampling techniques employed. Furthermore, there are most likely forums on the internet, apart from Wikipedia, where users converse in dialected Sardinian. While the text collected from such sources may not be as standard as found in a more formal setting, it may provide better insights for how people actually use the language, thus improving our model's robustness. Additionally, exploring social media platforms, blog posts, and online news articles written in Sardinian dialects could further enrich the training dataset.

If collecting new data is too difficult or costly, it could be fruitful to explore data augmentation techniques as described in Mohammed et al. (2020). These techniques include techniques such as character and word-level swapping, deletion, and insertion, which could help artificially expand the dataset and improve the model's performance on these variants.

Another improvement to this model would be to allow it to take as an input the variant. This in turn may produce more curated results for users who are already aware of the variant they are using, while potentially sacrificing training time. If we are already thinking about modifying the architecture, it could be of interest to modify it further. For instance, employing a CNN could perform better on the variant-prediction task. As mentioned earlier, replacing the LSTM with a GRU could also increase the speed of the model on lower-end devices while still performing similarly well (Cho et al., 2014).

While there was some hyper-parameter testing, the device used to train the model left much to be desired on this front. An obvious improvement to this model would be to test more hyperparameters for more epochs to find the best combination. To go along with this, testing different attention mechanisms could allow for better capturing of long-term dependencies.

Finally, the end goal would be to implement this model into a keyboard that could be used on mobile devices running Android or iOS. This would allow for real-world robust testing of the prediction algorithm. Furthermore, if data from users of the keyboard could be collected, it could increase the amount of training data available and in turn produce better and/or more personalized results. Personalization of predictive keyboards is not a novel idea, and has been explored by Microsoft (2023) and their Swiftkey Keyboard. To ensure user privacy and data security, it would be essential to implement proper anonymization and data handling practices, as well as to seek user consent before collecting any personal data for model improvement purposes.

References

2006. *Limba Sarda Comuna - Norme linguistiche di riferimento a carattere sperimentale per la lingua scritta dell'Amministrazione regionale*. Regione Autonoma della Sardegna.

Noëmi Aepli, Antonios Anastasopoulos, Adrian Chifu, William Domingues, Fahim Faisal, Mihaela Gaman Radu Tudor Ionescu, and Yves Scherrer. 2022. Findings of the vardial evaluation campaign 2022. *Ninth Workshop on NLP for Similar Languages, Varieties and Dialects*, pages 1–13.

Ahmed Ali, Stephan Vogel, and Steve Renals. 2017. https://doi.org/10.1109/ASRU.2017.8268952 Speech recognition challenge in the wild: Arabic mgb-3. In *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 316–322.

Héctor Alonso and Barbara Plank. 2020. Generating datasets for low-resource languages: A case study in language modeling. *arXiv preprint arXiv:2005.00722*.

Atelach Alemu Ayana and Wolfgang Menzel. 2017. Building a text corpus for representing the variety of language use across the peoples of ethiopia. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

Nikhil Barhate. 2019. `https://github.com/nikhilbarhate99/Char-RNN-PyTorch` Minimal implementation of multi-layer recurrent neural networks (lstm) for character-level language modelling in pytorch.

Jeremie Boudreau, Akankshya Patra, Ashima Suvarna, and Paul Cook. 2020. Evaluating the impact of subword information and cross-lingual word embeddings on mi'kmaq language modelling. *12th Language Resources and Evaluation Conference*, pages 2736–2745.

Rich Caruana. 1997. Multitask learning. *Machine Learning*, 28:41–75.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Jan K. Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. 2015. Attention-based models for speech recognition. *Advances in Neural Information Processing Systems*, 28:577–585.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Antonio Ciccolella. 2015. https://upload.wikimedia.org/wikipedia/commons/3/32/Dialetti_e_lingue_in_Italia.png Italiano: Mappa delle lingue e gruppi dialettali italiani.

Mereu D. 2019. https://doi.org/doi:10.1017/s0025100318000385 Cagliari sardinian. *Journal of the Internation Phonetic Associating*, pages 1–17.

Stephen L. Dyson and Robert J. Rowland. 2007. *Archaeology and history in Sardinia from the stone age to the Middle Ages: Shepherds, sailors, amp; conquerors*. University of Pennsylvania, Museum of Archaeology and Anthropology.

Alex Graves. 2013. https://doi.org/arXiv:1308.0850 Generating sequences with recurrent neural networks. *arXiv preprint*.

Liang Huang, Kai Zhao, and Migho Ma. 2018. https://doi.org/arXiv:1809.00069 When to finish? optimal beam search for neural text generation (modulo beam size). *EMNLP*.

Aaron Jaech, George Mulcaire, Mari Ostendorf, and Noah A. Smith. 2016. A neural model for language identification in code-switched tweets. In *CodeSwitch@EMNLP*.

Joo-Kyung Kim, Young-Bum Kim, Ruhi Sarikaya, and Eric Fosler-Lussier. 2016. Cross-lingual transfer learning for pos tagging without cross-lingual resources. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1064–1069.

Paul M. Lewis, editor. 2009. *Ethnologue: Languages of the World*, sixteenth edition. SIL International, Dallas, TX, USA.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. https://doi.org/arXiv:1605.05101v1 Recurrent neural network for text classification with multi-task learning. *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*.

Giovanni Lupinu, Alessandro Mongili, Anna Oppo, Riccardo Spiga, Sabrina Perra, and Matteo Valdes. 2007. `http://www.regione.sardegna.it/documenti/1_4_20070510134456.pdf` Le lingue dei sardi. una ricerca sociolinguistica.

Microsoft. 2023. Swiftkey keyboard. `https://www.microsoft.com/en-us/swiftkey`. Accessed: 4/23/2023.

Roweida Mohammed, Jumanah Rawashdeh, and Malak Abdullah. 2020. Machine learning with oversampling and undersampling techniques: Overview study and experimental results. *2020 11th International Conference on Information and Communication Systems (ICICS)*, pages 243–248.

Damien Mooney. 2017. *Creating Orthographies for Endangered Languages*. Cambridge University Press.

Christopher Moseley. 2010. *Atlas of the World's Languages in Danger*. UNESCO Publishing.

Seol-Hyun Noh. 2021. Analysis of gradient vanishing of rnns and performance comparison. *Information*, 12:1–11.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andres Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. `https://pytorch.org` Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems 32*, pages 8024–8035.

Bindu K. R., Akash Chan, Bernett Orlando, and Latha Parameswaran. 2018. https://doi.org/10.1007/978-3-319-71767-8_15 An algorithm for text prediction using neural networks. *Lecture Notes in Computational Vision and Biomechanics*, pages 186–192.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. https://doi.org/arXiv:1511.06732 Sequence level training with recurrent neural networks. *ICLR*.

Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*.

M. Sundermeyer, I. Oparin, J.-L. Gauvain, B. Freiberg, R. Schlüter, and H. Ney. 2013. https://doi.org/10.1109/ICASSP.2013.6639310 Comparison of feedforward and recurrent neural network language models. pages 8430–8434.

Martin Sundermeyer, Ralf Schluter, and Hermann Ney. 2012. Lstm neural networks for language modeling. *International Symposium on Computer Architecture 13th Annual Conference*, pages 194–197.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. *Advances in Neural Information Processing Systems*, 27:3104–3112.

Lei Tai and Ming Liu. 2016. Deep-learning in mobile robotics - from perception to control systems: A survey on why and why not. *Journal of LATEX class files*, 14:3.

Heidi Jauhiainen Tommi Jauhiainen and Krister Lindén. 2022. Italian language and dialect identification and regional french variety detection using adaptive naive bayes. *the Ninth Workshop on NLP for Similar Languages, Varieties and Dialects*, pages 119–129.

Ke Tran, Arianna Bisazza, and Christof Monz. 2018. Adapting word embeddings to new languages with morphological and phonological subword representations. *arXiv preprint arXiv:1808.06565*.

Stefania Tufi. 2013. https://doi.org/doi:10.1515/ijsl-2013-0009 Language ideology and language maintenance: The case of sardinia. *International Journal of the Sociology of Language*, 2013(219).

K. Vertanen, H. Memmi, J. Emge, S. Reyal, and P. O. Kristensson. 2015. https://doi.org/10.1145/2702123.2702135 Velocitap: Investigating fast mobile text entry using sentence-based decoding of touchscreen keyboard input. pages 659–668.

Keith Vertanen and Per Ola Kristensson. 2011. https://doi.org/10.1145/2037373.2037418 A versatile dataset for text entry evaluations based on genuine mobile emails. page 295–298, New York, NY, USA. Association for Computing Machinery.

Max W. Wheeler. 2005. *The Phonology of Catalan*. Oxford University Press, Oxford.

S. Zhai, A. Sue, and J. Accot. 2002. Movement model, hits distribution and learning in virtual keyboarding. *SIGCHI conference on Human factors in computing systems*, pages 17–24.

Min Zhang and Juntao Li. 2021. https://doi.org/10.1016/j.fmre.2021.11.011 A commentary of GPT-3 in MIT Technology Review 2021. *Fundamental Research*, 1(6):831–833.