Diacritic Restoration for Vietnamese

Daniel Saelid[♠] **Sachin Kumar**[♣] **Yulia Tsvetkov**[♠]
[♠]Paul G. Allen School of Computer Science & Engineering, University of Washington, Seattle WA
[♣]Language Technologies Institute, Carnegie Mellon University, Pittsburgh PA
saeliddp@cs.washington.edu, sachink@cs.cmu.edu, yuliats@cs.washington.edu

Abstract

Vietnamese orthography relies on Latin characters augmented with diacritic marks. Vietnamese is often written on digital systems without diacritic marks since it is difficult to add diacritics in a world where most keyboards are based on ASCII. While a fluent Vietnamese reader would understand most ASCII-ified Vietnamese passages from context, downstream systems such as search engines may not. Diacritic restoration for Vietnamese is thus necessary to ensure that all Vietnamese speakers can use computer systems successfully. This paper describes experiments with generative and deep learning models for diacritic restoration. All code is available at a public GitHub repository.[1] The highest performing model is published on HuggingFace. [2]

1 Introduction

Standard Vietnamese is written with 22 Latin characters–all except *f*, *j*, *w*, and *z*. Seven additional characters are formed by adding a diacritic mark to a standard Latin character: *ă, â, đ, ê, ô, ơ, ư*. Since Vietnamese is a tonal language, every syllable in a phrase must be pronounced with exactly one of six tones; otherwise, the meaning of the phrase will change. In writing, five of the tones are represented as diacritic marks: the rising tone is represented with the acute (*á*), the falling tone with the grave (*à*), the falling-rising tone with the hook (*ả*), the high-cracking tone with the tilde (*ã*), and the stopping tone with the dot below (*ạ*). The sixth, flat tone is represented by the absence of all other tonal diacritics (*a*). Tone marks are placed above one of the vowels in a syllable.

The problem of diacritic restoration for Vietnamese is as follows: given Vietnamese text written with only ASCII characters (Vi-ASCII), add diacritics where appropriate to form valid written Vietnamese (Vi-Std). For instance, if the Vi-ASCII sentence was 'Minh dang di an', the correct restoration would be 'Mình đang đi ăn' (we're going out to eat). Note that in some cases, multiple correct restorations are possible for the text. To illustrate, 'dua moc o dau?' could be restored to either 'dứa mọc ở đâu?' ('where do pineapples grow?') or 'dừa mọc ở đâu?' ('where do coconuts grow?'). So long as reasonable Vietnamese is produced by the restoration, the restoration is successful. The question of what counts as 'reasonable' is interesting, but beyond the scope of this paper. For the experiments in this paper, we strip diacritics from Vi-Std text, attempt to restore them, and consider the restoration successful if the restored diacritics match the original diacritics.

In this paper, we build three models for diacritic restoration. The baseline model, MFreq, simply chooses the most frequently occurring Vi-Std form of a Vi-ASCII syllable. The HMM model (Eisenstein, 2019) treats the problem as sequence labeling. The tokens (x) are Vi-ASCII syllables, while the labels (y) are Vi-Std syllables. The hidden Markov model then calculates $\text{argmax}_{y_1 \ldots y_n} \Pi_{i=1}^{n} P(x_i|y_i) \cdot P(y_i|y_{i-1})$ and treats this sequence of labels as the Vi-Std form of the input sentence. Finally, the DistilBERT model (Sanh et al., 2020) also treats the problem as sequence labeling, but the tokens are individual characters from a Vi-ASCII sentence, while the labels are diacritic marks. Upon receiving the input $c_1 \ldots c_n$, DistilBERT will choose the diacritic mark with the highest probability for each c_i. We find that HMM performs nearly as well as DistilBERT, but is prone to make many mistakes once a single mistake is made in a sentence restoration. For detailed results, see section 6.

[1] saeliddp/DRST
[2] saeliddp/distilbert-viet-diacritic-restoration

2 Motivation

The difference between Vi-Std and Vi-ASCII is significant. According to Stankevičius et al. (2022), 25.95% of Vietnamese and 81.18% of Vietnamese syllables contain diacritic marks.

In order to type in Vi-Std, one must rely on an input method such as TELEX,[3] which translates ASCII keystrokes from a QWERTY keyboard into Vietnamese characters. For instance, if one types 'hoom nay laf thuws bary' into a system using TELEX, 'hôm nay là thứ bảy' ('today is Saturday') would appear on the screen. TELEX leverages Vietnamese orthographic and phonological restrictions (e.g. *w* is never used in Vietnamese, *s* never appears at the end of a syllable) to form an unambiguous encoding.

However, not every Vietnamese speaker knows how to use TELEX or alternatives.[4] Moreover, even those who know TELEX may opt to write Vi-ASCII for convenience. Vi-ASCII is usually sufficient for communication purposes, as the reader of the text will likely be able to infer diacritics from context.

While diacritic restoration is not crucial for direct communication between Vietnamese speakers through computer systems, it becomes important when Vietnamese speakers wish to leverage advanced NLP technologies.

As a case study, suppose a user of a blogging website remembers reading a useful post about how to cut pineapples, so she opens the blogging website's search feature and types 'cách cắt dứa' ('how to cut pineapple') using TELEX. After scrolling through a few pages of results, she can't find the blog post, and she remembers that the post was written without diacritics. So, she changes her search query to 'cach cat dua.' Now, she is presented with a mixture of articles about cutting pineapples ('dứa'), coconuts ('dừa'), and melons ('dưa'). She has to carefully search through many pages of results to find the post she remembers reading.

The only way the blogging website could address the user's problem would be to introduce a diacritic restoration system for their search engine. If they could index posts based on their original text content and the text content with diacritics restored, then the user's original query 'cách cắt dứa' would return the result she was looking for.

From the above case study, it's clear that diacritic restoration can be a necessity. In other circumstances, it may just improve a user's experience if they can quickly type Vi-ASCII and see the Vi-Std that they would write by hand show up on their screen instead.

3 Related Work

The earliest approaches for diacritic restoration replaced ASCII tokens with their most frequent Std form (Yarowsky, 1999). While effective for high-resource languages which sparingly use diacritics, this approach does not generalize to low-resource languages or languages like Vietnamese which extensively use diacritics. De Pauw et al. (2007) designed an alternative approach which they hoped would be suitable for resource-scarce languages. They designed a character classifier which takes a window of surrounding ASCII characters as input features and predicts the diacritic mark for the central character. This technique offered only a slight performance improvement on Vietnamese.

One of the first methods to significantly improve upon the most frequent model was designed by Truyen et al. (2008). The authors proposed a 'Powered Product-of-N-gram' model, which predicts diacritics for a sentence by choosing the sequence of Std forms of ASCII syllables which maximizes a weighted product of multiple N-gram probabilities for that sentence.

Recently, the most successful diacritic restoration systems have been built with deep learning architectures. Some researchers have used machine translation models to translate from ASCII to standard 'languages' (Stankevičius et al., 2022; Pham et al., 2017; Hung, 2018). Others have used RNNs (Hucko and Lacko, 2018), CNNs (Alqahtani et al., 2019), and BERT models (Náplava et al., 2021).

To the best of our knowledge, the current state-of-the-art system for Vietnamese diacritic restoration achieves an alpha-word accuracy of 98.53% by fine tuning a pre-trained multilingual BERT model for the task of diacritic restoration (Náplava et al., 2021).

4 Methodology

In this paper, we build and examine three models: MFreq, HMM, and DistilBERT. Each is described below.

MFreq: During training, we calculate Vi-Std frequencies for each Vi-ASCII token. For each unique

[3] Telex (input method)

[4] History of Vietnamese typing

Vi-ASCII syllable x_k, we record the frequency of each Vi-Std syllable it is mapped to in the training data. At the end of training, we have a function $count(x, y)$ which will return the number of times the Vi-ASCII syllable x was mapped to the Vi-Std form y.

During inference, for a given Vi-ASCII syllable x_k, we predict the Vi-Std form to be $\text{argmax}_y\, count(x_k, y)$.

HMM: During training, we calculate emission probabilities ($P(x_i|y_i)$, the probability of the Vi-ASCII syllable given the Vi-Std syllable) and transition probabilities ($P(y_i|y_{i-1})$, the probability of the current Vi-Std syllable given the previous Vi-Std syllable). We apply Laplace smoothing to the transition probabilities.

During inference, the input is a sentence $x_1 ... x_n$. We use constrained Viterbi decoding to find $\text{argmax}_{y_1 ... y_n} \Pi_{i=1}^{n} P(x_i|y_i) \cdot P(y_i|y_{i-1})$. The decoding is called constrained because at each step, we consider only the possible Vi-Std versions of the Vi-ASCII syllables (Truyen et al., 2008). For example, if the current token is 'co', we would consider 'có', 'cô', etc. but not 'trường' or 'sắp' even though the latter two are also in the set of Vi-Std labels. Without constrained decoding, the computation time for Viterbi would be prohibitive. Our HMM deviates from the canonical setup in another way: when we encounter a Vi-ASCII syllable which was not in our training data, we do not attempt to predict a Vi-Std form and instead leave it as-is. The reasoning here is that if we have never seen a Vi-ASCII syllable, then there is no possibility that we have a correct label (Vi-Std form) for it in our set of labels. Every attempted prediction will thus be incorrect, but leaving it as-is has a chance of being correct since some Vietnamese syllables contain no diacritic marks.

DistilBERT: We take Hugging Face's pre-trained 'distilbert-base-multilingual-cased' model (Sanh et al., 2020) and fine tune it for character-by-character diacritic restoration for Vietnamese. Our sequence classification model, DistilBERTForTokenClassification[5], takes the contextual character embeddings produced by the distilled BERT model, then runs them through a linear layer to get logits over possible diacritics for each character in the sequence. We train all parameters (both BERT and classification) with the AdamW optimizer. The learning rate reaches a maximum of 0.0001 after linearly increasing for 1000 warmup steps. We train for three epochs, which takes approximately 6 hours on a cluster of four NVIDIA RTX A4000.

During inference, we simply run our input of characters $c_1 ... c_n$ through the model. We mark each character with the diacritic mark corresponding to the maximum value in the output logit for that character.

[5] HuggingFace class

Split	Sentences	Syllables
Train	819,918	21,379,153
Development	14,884	441,152
Test	30,000	841,328

Table 1: Split sizes.

5 Experimental Setup

We use the Vietnamese portion of the diacritic restoration corpus compiled by Náplava et al. (2018). Specifically, we utilize the pre-split Wikipedia training, development, and test sets. This text data comes lightly pre-processed: there is one sentence per line, all characters are lowercase, and punctuation is separated from syllables with whitespace. The sizes of the splits are shown in Table 1. The text data is all written in Vi-Std (with diacritics). So, in order to form the source (Vi-ASCII) sentences, we simply strip diacritic marks from the Vi-Std sentences. Throughout our experiments, three tokenization/labeling schemes are employed. **syll-syll** considers a list of Vi-ASCII syllables as input and a list of Vi-Std syllables as labels. **char-char** considers a list of Vi-ASCII characters as input and a list of Vi-Std characters as labels. **char-diac** considers a list of Vi-ASCII characters as input and a list of diacritic marks as labels. The results of running each on the sentence 'có rồi' are shown below.

1. **syll-syll**:
   - input: [SENT-START, co, roi]
   - labels:[SENT-START, có, rồi]
2. **char-char**:
   - input: [WORD-START, c, o, WORD-START, r, o, i]
   - labels: [WORD-START, c, ó, WORD-START, r, ồ, i]
3. **char-diac**:

- input: [WORD-START, c, o, WORD-START, r, o, i]
- labels: [NONE-NONE, NONE-NONE, NONE-ACUTE, NONE-NONE, NONE-NONE, TAIL-ACUTE, NONE-NONE]

For all tokenization schemes, if a syllable in the original text contains a numeral, the syllable will be collapsed down to a special NUMERIC token. Furthermore, all syllables which contain no alphabetic characters are ignored. Additionally, **syll-syll** uses the special SENT-START token, while **char-char** and **char-diac** use the special WORD-START token.

We define three accuracy measures for evaluation: character, syllable, and sentence accuracy. Character accuracy is the percent of characters restored correctly, syllable accuracy is the percent of syllables restored correctly, and sentence accuracy is the percent of sentences restored correctly. If a single syllable in a sentence is restored incorrectly, then the sentence is restored incorrectly. Similarly, if a single character in a syllable is restored incorrectly, then the syllable is restored incorrectly. Each of these measures ignores any special characters present in the data. So, these are all measures of accuracy if we ignore all non-strictly-alphabetic words. As such, syllable accuracy here is equivalent to alpha-word accuracy from (Náplava et al., 2021).

6 Results and Analysis

6.1 Accuracy

MFreq: Accuracies for most frequent model.

Tokenizer	Character	Syllable	Sentence
syll-syll	91.98	73.56	2.85
char-char	73.75	15.43	0.67
char-diac	74.34	17.42	0.68

Table 2: MFreq accuracies (in %).

HMM: Accuracies for hidden Markov model.

Tokenizer	Character	Syllable	Sentence
syll-syll	97.93	93.46	35.51
char-char	84.44	49.59	0.90
char-diac	75.59	20.12	0.75

Table 3: HMM accuracies (in %).

DistilBERT: Accuracies for distilled BERT model.

For the DistilBERT model, we only ran experiments with the **char-diac** tokenizer. There are a few reasons for this. First, note that for DistilBERT, the problem of predicting diacritic marks for each of the Vi-ASCII characters in the input sequence is equally as hard as predicting Vi-Std characters. This is because, unlike an HMM or RNN, the transformer architecture does not use the predicted labels for other characters in the sequence when predicting the label for a given character (Devlin et al., 2019). So, we expect **char-char** and **char-diac** to yield similar results, but **char-diac** uses fewer parameters. Additionally, we chose not to experiment with **syll-syll** because the number of 'classes' is prohibitive. If we used **syll-syll**, then we would be attempting a classification problem with as many classes as there are tokens in the Vi-Std vocabulary. Presumably, the necessary training time for such a model would be extreme.

Tokenizer	Character	Syllable	Sentence
syll-syll	N/A	N/A	N/A
char-char	N/A	N/A	N/A
char-diac	98.75	96.10	50.26

Table 4: DistilBERT accuracies (in %).

Model	Char	Syll	Sent
MFreq/syll-syll	91.98	73.56	2.85
HMM/syll-syll	97.93	93.46	35.51
DistilBERT/char-diac	98.75	96.10	50.26

Table 5: Accuracies for most performant version of each model (in %).

6.2 Analysis

For both MFreq and HMM, **syll-syll** tokenization performs the best. For MFreq, predicting the most frequent Vi-Std character for a given Vi-ASCII character is equivalent to predicting the most frequent diacritic mark for that Vi-ASCII character. As such, the accuracies for **char-char** and **char-diac** are identical for MFreq. The same is not true for the HMM. This is because knowing the previous Vi-Std character gives the model more information than knowing only the previous diacritic mark. For instance, in Vi-Std, 'q' is always followed with 'u'. The **char-char** HMM will estimate the probability of the current character being 'u' given that the pre-

vious character was ‘q’, but the **char-diac** HMM will only estimate the probability of the current character having no diacritics given that the previous character did not have diacritics. As such, the transition probabilities are more useful in the **char-char** HMM which increases accuracy significantly.

One explanation for why DistilBERT outperforms HMM is that it is able to use more context when predicting diacritic marks. A simple test for this hypothesis involves chunking the input sentences into blocks of n syllables, then running each chunk through the DistilBERT model separately. By altering n, we are able to restrict the context that DistilBERT has access to.

Syllables in Chunk (n)	Syllable Accuracy
2	76.78
4	87.64
8	92.38
16	94.56
32	95.69
64	96.07

Table 6: DistilBERT word accuracies (in %) based on chunk size.

From the data in table 6, DistilBERT begins to outperform HMM when the number of syllables in the chunk reaches 16. The positive correlation between chunk size and syllable accuracy supports the hypothesis that DistilBERT outperforms HMM due to its effective use of distant context.

For the MFreq model, the source of error is clear: MFreq will always predict the most common Vi-Std version of a Vi-ASCII syllable, so uncommon Vi-Std syllables will never be predicted correctly. We formalize the idea of ‘commonness’ in relative frequency as follows. Suppose that the Vi-ASCII syllable s_A can be diacritized to form the Vi-Std syllables $s_1 ... s_n$. Then, the relative frequency of s_k is $RelFreq(s_k) = \frac{count(s_k)}{\sum_{i=1}^{n} count(s_i)}$ where $count(s_i)$ is the number of times s_i appears in the training corpus.

While DistilBERT is clearly not always predicting the most common Vi-Std version of a Vi-ASCII syllable, it may still perform worse when the correct Vi-Std syllable is a relatively infrequent one. Figures 1 and 2 show recall and precision for Vi-Std syllables by relative frequency buckets.

Clearly, recall decreases as relative frequency decreases. So, when DistilBERT should predict a relatively infrequent syllable, it tends to consistently predict other, more relatively frequent Vi-Std versions of the same Vi-ASCII syllable.

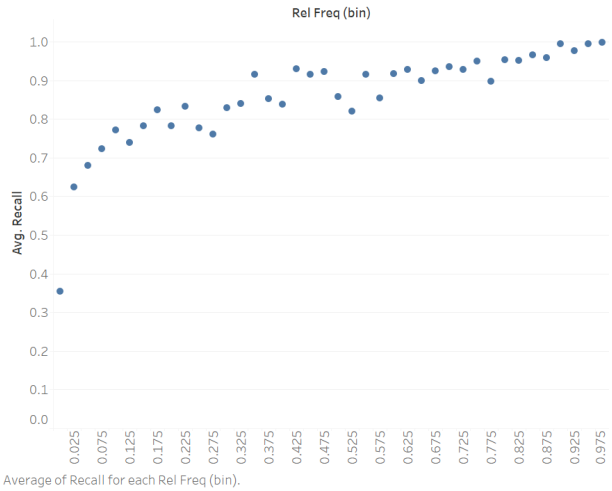

Figure 1: Recall for Vi-Std syllables based on relative frequency

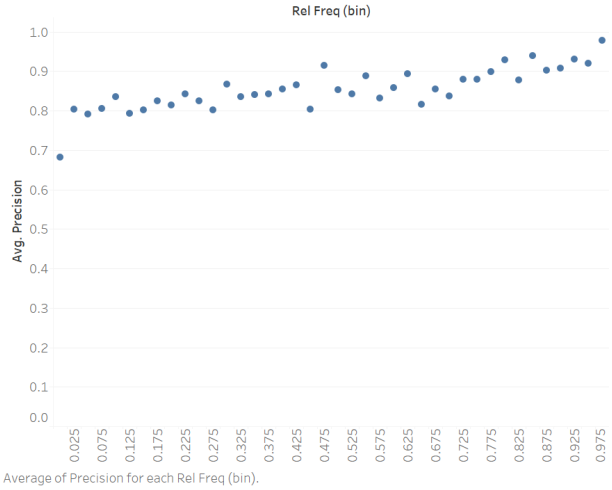

Figure 2: Precision for Vi-Std syllables based on relative frequency

Precision also decreases as relative frequency decreases, but it does so at a slower rate than recall. This means that DistilBERT will generally not predict a relatively infrequent Vi-Std version of a Vi-ASCII syllable unless it is very clear that the Vi-ASCII syllable should be restored to the relatively infrequent Vi-Std version.

This relative frequency analysis on syllables should imply that if a sentence contains a relatively infrequent syllable, then correct restoration is less likely for the sentence as a whole. Figure 3 verifies this intuition.

With an average sentence length of 27.9 syllables and a median sentence length of 25 sylla-

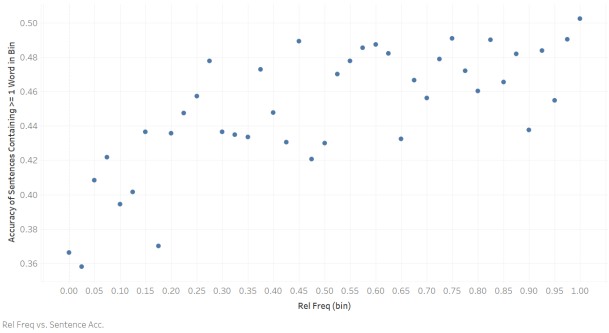

Figure 3: Accuracies for all sentences that contain a syllable in the given relative frequency bin.

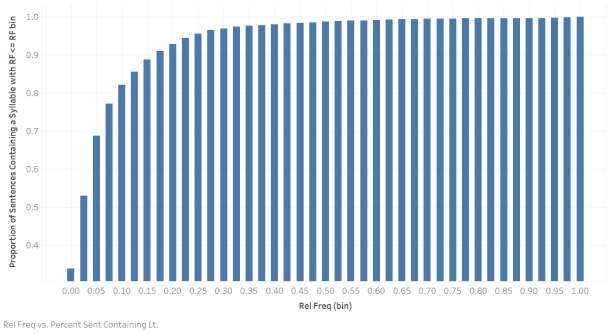

Figure 4: Proportion of sentences containing Vi-Std syllables with relative frequency less than or equal to the relative frequency bin.

bles, we would expect the sentence accuracy for DistilBERT to be $0.9610^{26} = 0.3555$ (this is a rough estimate). However, the sentence accuracy is measured at 0.5026. This implies that when the model makes a mistake in one sentence, it tends to make many mistakes in that sentence. In other words, it seems to perform quite poorly on the sentences where it it does not perform perfectly (49.74% of the sentences). If we conduct a similar analysis for HMM, we find an even greater difference between the expected and observed sentence accuracies ($expected = 0.9346^{26} = 0.1723$, $observed = 0.3551$). This suggests that the HMM is less robust than DistilBERT when 'confusing' syllables are present in the sentence.

Another possible interpretation of this data is that most of the sentences contain only relatively frequent Vi-Std syllables. However, figure 4 suggests otherwise. Over 50% of sentences contain syllables with relative frequencies ≤ 0.025, and over 90% of sentences contain syllables with relative frequencies ≤ 0.575. A thorough analysis to explain sentence accuracy's relationship with syllable accuracy would require more sophisticated visualizations which are beyond the scope of this paper.

7 Conclusion and Future Work

In this paper, we built and evaluated three models for diacritic restoration for Vietnamese. DistilBERT achieved the highest accuracy because it was able to attend to distant context from the input sentence when deciding which diacritic marks to add. HMM also performed well, lagging only a few percentage points behind DistilBERT for syllable accuracy. Thus, in many situations, the simpler HMM model will be sufficient for diacritic restoration tasks.

To the best of our knowledge, prior work on diacritic restoration has focused on reporting only syllable/word accuracy and character accuracy. This paper includes sentence accuracy as an additional metric and shows how doing so may lead to productive analysis.

Future work for the Vietnamese diacritic restoration task can take many directions. First, we could train our DistilBERT model for longer to test whether a distilled BERT architecture can achieve equivalent accuracy to a standard BERT architecture (Náplava et al., 2021) given enough training data. Additionally, we could evaluate our model on more natural data sources like blog posts. Another interesting direction would be to build and test simultaneous diacritic restoration systems that insert diacritic marks as the user types in Vi-ASCII (Cho and Esipova, 2016).

Finally, existing work does not address the problem of true ambiguity for diacritic restoration. In a nutshell, even if the system restores diacritics and produces a completely sensible sentence, it will be counted as a failure if the original sentence had an alternate meaning (and thus different diacritic marks). It would be useful to know how much this problem affects accuracy measures; if the effect is significant, solutions are necessary.

References

Sawsan Alqahtani, Ajay Mishra, and Mona Diab. 2019. Efficient convolutional neural networks for diacritic restoration. page 1442–1448, Hong Kong, China. Association for Computational Linguistics.

Kyunghyun Cho and Masha Esipova. 2016. Can neural machine translation do simultaneous translation? (arXiv:1606.02012). ArXiv:1606.02012 [cs].

Guy De Pauw, Peter W. Wagacha, and Gilles-Maurice

de Schryver. 2007. *Automatic Diacritic Restoration for Resource-Scarce Languages*, volume 4629 of *Lecture Notes in Computer Science*, page 170–179. Springer Berlin Heidelberg, Berlin, Heidelberg.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. (arXiv:1810.04805). ArXiv:1810.04805 [cs].

Jacob Eisenstein. 2019. *7.4 Hidden Markov Models*, page 145–149. The MIT Press.

Andrej Hucko and Peter Lacko. 2018. Diacritics restoration using deep neural networks. In *2018 World Symposium on Digital Intelligence for Systems and Machines (DISA)*, page 195–200, Kosice. IEEE.

Bui Thanh Hung. 2018. Vietnamese diacritics restoration using deep learning approach. In *2018 10th International Conference on Knowledge and Systems Engineering (KSE)*, page 347–351, Ho Chi Minh City. IEEE.

Jakub Náplava, Milan Straka, Jan Hajič, and Pavel Straňák. 2018. Corpus for training and evaluating diacritics restoration systems. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Jakub Náplava, Milan Straka, and Jana Straková. 2021. Diacritics restoration using bert with analysis on czech language. *Prague Bulletin of Mathematical Linguistics*, 116(1):27–42.

Thai-Hoang Pham, Xuan-Khoai Pham, and Phuong Le-Hong. 2017. On the use of machine translation-based approaches for vietnamese diacritic restoration. In *2017 International Conference on Asian Language Processing (IALP)*, page 272–275, Singapore. IEEE.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. (arXiv:1910.01108). ArXiv:1910.01108 [cs].

Lukas Stankevičius, Mantas Lukoševičius, Jurgita Kapočiūtė-Dzikienė, Monika Briedienė, and Tomas Krilavičius. 2022. Correcting diacritics and typos with a byt5 transformer model. *Applied Sciences*, 12(5):2636.

Tran The Truyen, Dinh Q. Phung, and Svetha Venkatesh. 2008. *Constrained Sequence Classification for Lexical Disambiguation*, volume 5351 of *Lecture Notes in Computer Science*, page 430–441. Springer Berlin Heidelberg, Berlin, Heidelberg.

D. Yarowsky. 1999. *A Comparison of Corpus-Based Techniques for Restoring Accents in Spanish and French Text*, volume 11 of *Text, Speech and Language Technology*, page 99–120. Springer Netherlands, Dordrecht.