Data Augmentation for Maltese NLP using Transliterated and Machine Translated Arabic Data

Kurt Micallef[1,2] kurt.micallef@um.edu.mt

Nizar Habash[2] nizar.habash@nyu.edu

Claudia Borg[1] claudia.borg@um.edu.mt

[1]Department of Artificial Intelligence, University of Malta
[2]Computational Approaches to Modeling Language Lab, New York University Abu Dhabi

Abstract

Maltese is a unique Semitic language that has evolved under extensive influence from Romance and Germanic languages, particularly Italian and English. Despite its Semitic roots, its orthography is based on the Latin script, creating a gap between it and its closest linguistic relatives in Arabic. In this paper, we explore whether Arabic-language resources can support Maltese natural language processing (NLP) through cross-lingual augmentation techniques. We investigate multiple strategies for aligning Arabic textual data with Maltese, including various transliteration schemes and machine translation (MT) approaches. As part of this, we also introduce novel transliteration systems that better represent Maltese orthography. We evaluate the impact of these augmentations on monolingual and mutlilingual models and demonstrate that Arabic-based augmentation can significantly benefit Maltese NLP tasks.

1 Introduction

Maltese is the only Semitic language written in the Latin script and the only one that is an official language of the European Union. It retains a core Semitic structure but has undergone extensive lexical borrowing and structural influence from Italian and English. Despite its roots in North African Arabic, modern Maltese and Arabic are now orthographically and lexically distant, posing unique challenges for leveraging Arabic NLP resources to support Maltese.

Given the low-resource status of Maltese, recent work has focused on leveraging multilingual language models and transfer learning (Lauscher et al., 2020; inter alia). In this paper, we pursue a complementary strategy: enriching Maltese datasets through augmentation derived from Arabic resources. While doing so, we address the divergence in script and phonology between Arabic and Maltese, by considering multiple layers of transliteration and translation to Arabic inputs. These include transliteration schemes from previous works such as Buckwalter (Buckwalter, 2002), Uroman (Hermjakob et al., 2018), as well as machine translations. As part of this work, we develop novel phonology- and morphology-aware transliteration systems, which we publicly release.[1] See example in Table 1. We apply these augmentations individually and in combination.

Arabic	أوقفت السيارة في الطريق.
Buckwalter	>wqft AlsyArp fy AlTryq.
Uroman	awqft alsyara fy altryq.
CharTx	uqft alsjara fi altriq.
MorphTx	awqefat is-sejjara fit-teriq.
MT	Waqqaft il-karozza fit-triq.
Maltese	Ipparkjajt il-karozza fit-triq.
English	I parked the car on the street.

Table 1: Arabic sentence and its Buckwalter, Uroman, and transliterations using our new systems (CharTx and MorphTx), along with Maltese machine translation (MT) and native Maltese versions.

To assess the effectiveness of Arabic augmentation, we evaluate its impact across three language models: **mBERT**, **BERTu** (a Maltese BERT model), and **mBERTu** (mBERT with additional Maltese pre-training). Our experiments demonstrate that the type of augmentation data significantly affects downstream performance and that models with more Maltese knowledge benefit differently from Arabic augmentation compared to less Maltese-aware models. We also show that combining multiple augmentation techniques is helpful.

The paper presents related work in Section 2, describes our transliteration methods in Section 3, outlines the experimental setup in Section 4, and presents our results in Sections 5 and 6.

[1] https://www.github.com/MLRS/maltify_arabic

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 21580–21590
November 4-9, 2025 ©2025 Association for Computational Linguistics

2 Related Work

2.1 Cross-lingual Transfer

Multilingual models have been shown to be quite effective for cross-lingual transfer (Wu and Dredze, 2019), but their effectiveness on low-resource languages is often limited by their representation in the model's pre-training data, which is often small or non-existent (Wu and Dredze, 2020; Lauscher et al., 2020; Muller et al., 2021; Winata et al., 2022). Low-resource languages are particularly limited to generalise well due to miniscule annotated datasets. However, few-shot cross-lingual fine-tuning has been shown to be an effective strategy, where data from a high-resource language is used to improve performance on a low-resource language (Lauscher et al., 2020; Zhao et al., 2021; Schmidt et al., 2022).

Previous works have shown that script mismatch can particularly impact performance, while transliteration can be used as a way to align languages in a unified script and boost cross-lingual transfer (Muller et al., 2021; Liu et al., 2025). Closer to our setting, Micallef et al. (2023) study Maltese as a dialect of Arabic through transliteration. We extend this by comparing broader data augmentation strategies and analysing their effects across models with varying degrees of Maltese similarity.

2.2 Transliteration

Various Arabic-to-Latin transliteration systems have been proposed with differing goals. Buckwalter maps Arabic letters to ASCII characters deterministically (Buckwalter, 2002), while CAPHI emphasizes phonetic accuracy (Habash et al., 2018). Uroman offers general Latin-script mappings for many languages, including Arabic (Hermjakob et al., 2018). Eryani and Habash (2021) propose a Romanisation system tailored for diacritised bibliographic records. We use Buckwalter and Uroman as baselines in our experiments, but note that these representations often diverge from Maltese orthography, limiting their effectiveness for our task.

3 Our Transliteration Systems

In this section, we outline the two Arabic-to-Maltese transliteration systems we developed, with all mapping rules listed in Appendix A.

3.1 CharTx: Character Mappings

We define Arabic-to-Maltese character mappings by reversing the Maltese-to-Arabic rules from Micallef et al. (2023). Ambiguities in that work, such as 's' mapping to س *s* or ص *S*, collapse trivially to one Maltese form in our case. More challenging are the Arabic glides ي *y* and و *w*, which can function as long vowels when preceded by ◌ِ *i* and ◌ُ *u* diacritics, respectively (Habash, 2010). We handle these by enumerating all diacritic combinations and mapping accordingly.

Our system avoids generating Maltese letters like 'ċ', 'g', 'p', 'v', and 'z', as these largely arise from non-Arabic sources. For instance, while 'g' could map from ج *j*, we consistently map it to 'ġ', its more frequent counterpart. Letters like 'p' and 'v' may relate to ب *b* or ف *f*, but are rarely found in words of Arabic origin.

Finally, characters not explicitly mapped are preserved. The mappings are applied at the word level, so we tokenise using CAMeL Tools when needed (Obeid et al., 2020). For cases where we tokenise we rejoin transliterated tokens and detokenise to remove unwanted spacing before punctuation.[2]

3.2 MorphTx: Morphological Features

On top of character mappings, we incorporate a linguistic disambiguator to predict diacritised forms using CAMeL Tools (Obeid et al., 2020), which implements the BERT-based model of Inoue et al. (2022) with the Egyptian CALIMA C044 morphological database (Habash et al., 2012). We use the Egyptian model due to the greater similarity of Maltese to Dialectal Arabic over MSA. Suitable analysers for closer dialects like Tunisian were not available. However, when predictions fail under the Egyptian model, the system falls back to MSA.

The disambiguator selects the highest-scoring diacritised form from all possible morphological analyses in context, enabling more accurate application of the character mappings (CharTx) from Section 3.1. It also provides morpheme segmentation and POS tags, allowing us to define morpheme-specific mappings that override character-level rules. For example, `DET` ال *Al* maps to 'il-', and `PRON_2MP` maps to 'kom'. We include mappings for all Arabic affixes in the analyser database, and map them appropriately to Maltese. We also capitalise morphemes tagged as proper nouns (`NOUN_PROP`). Additionally, following Micallef et al. (2023), we handle Maltese orthographic conventions such as the contraction of *fi il-* to *fil-* 'in the', and sun letter assimilation, e.g., *il-żejt* to *iż-żejt* 'the oil'.

[2] The original boundaries may not always be preserved.

4 Experimental Setup

Our goal is to improve cross-lingual transfer to Maltese by transliterating Arabic and using it for data augmentation. Our few-shot setup involves fine-tuning on Arabic data, followed by Maltese. Section 4.1 outlines the datasets used, and Section 4.2 details the models and input processing. Appendix C includes more technical details on our fine-tuning setup.

4.1 Datasets

Named-Entity Recognitions (NER) For Arabic, we use ANERCorp (Benajiba et al., 2007) with the splits from Obeid et al. (2020). For Maltese, we use the MAPA (Gianola et al., 2020) with the splits and fixes from Micallef et al. (2024). We normalise both datasets to a common tagset and also downsample the Maltese data. After doing so, we have 3,973 and 155 training sentences for Arabic and Maltese, respectively. Span-level F1 is used when evaluating model outputs.

Sentiment Analysis (SA) For Arabic we make use of the data from Baly et al. (2018), while for Maltese we use the data from Martínez-García et al. (2021). Since the Maltese data only has positive and negative sentences, we drop any neutral sentences from the Arabic data. After this filtering process we have 15,305 and 595 training sentences for Arabic and Maltese, respectively. We use macro-averaged F1 to evaluate the models.

All of the Arabic data is pre-processed using CAMeL Tools `arclean`, which normalises ambiguous Arabic characters, that could be potentially problematic in our modelling (Obeid et al., 2020). Appendix B includes more details on the datasets used, including our filtering and processing steps. The Arabic data is only used for training, and the validation and testing data are always Maltese. Table 2 provides a breakdown of the final dataset sizes that are used in this work.

4.2 Models and Inputs

We experiment with the following models:

- **BERTu** (Micallef et al., 2022): a monolingual model pre-trained on Maltese.
- **mBERT** (Devlin et al., 2019): a multilingual model that includes Arabic in its pre-training, but not Maltese.
- **mBERTu** (Micallef et al., 2022): mBERT further pre-trained on Maltese.

Dataset	Training	Validation	Testing
Named-Entity Recognition (NER)			
Arabic	3,973	-	-
Maltese	155	43	2,109
Sentiment Analysis (SA)			
Arabic	15,305	-	-
Maltese	595	85	171

Table 2: Data sizes for the downstream tasks

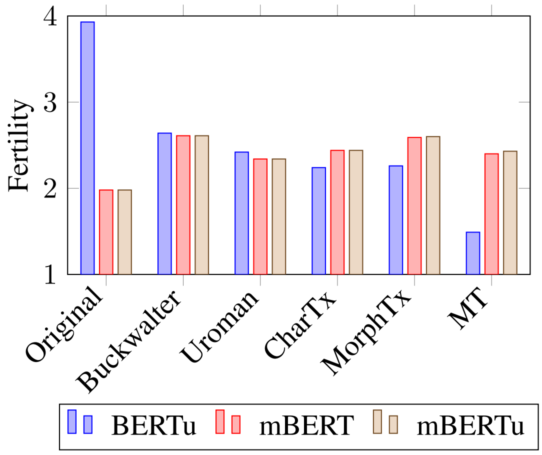

Figure 1: Tokeniser fertility across datasets using the different Arabic inputs and models.

In terms of Arabic inputs used for fine-tuning, we compare the original Arabic data in its original Arabic script (**Original**) with the different transliteration systems. In addition to our transliteration systems – **CharTx** and **MorphTx** – we compare against two generic transliteration systems: **Buckwalter** and **Uroman**. For Buckwalter we also lowercase the produced transliteration, since it uses uppercase letters for some of its mappings, while uppercase letters carry linguistic meaning in Latin-script languages.

We also include machine translation (**MT**) to measure the differences between translating and transliterating Arabic data. We do so using Google Translate, from Arabic into Maltese.

To analyse the impact of using different Arabic inputs, we compute the tokeniser fertility, which measures the average number of sub-tokens that the tokeniser splits a given token into (Ács, 2019). For each input, we compute the fertility when processed with a model's tokeniser and visualise this in Figure 1.

BERTu with Original Arabic data has a high fertility (close to 4), but significantly drops with any of the transliteration inputs, dropping even further with machine translation. We note that both our transliteration systems – CharTx and MorphTx – give lower fertility scores than the other transliteration systems, as we move closer towards Maltese orthography. Fertility dropps even further with machine translated data.

Conversely, for mBERT and mBERTu, the lowest fertility is obtained with Original and increases with transliteration and machine translation, reflecting the overall lack of Maltese pre-training and vocabulary representation with these models.

5 Transliteration Fine-Tuning Results

We first compare the different transliteration systems extrinsically with models fine-tuned on original Arabic as well as fine-tuning only on Maltese data. Results are shown in Table 3.

No Arabic vs Original Arabic When adding original Arabic data, BERTu shows performance drops as expected, since it is only pre-trained on Maltese. However, mBERT and mBERTu generally improve with Arabic data, though mBERTu slightly underperforms on SA.

No Arabic vs Transliterated Arabic Adding transliterated data shows mixed results depending on the model and transliteration system. Buckwalter and Uroman underperform on NER for BERTu and mBERTu, though Uroman helps on SA. Conversely, mBERT benefits from both Buckwalter and Uroman. Our transliteration systems improve BERTu on both NER and SA tasks and outperforms Buckwalter for mBERT and mBERTu, with CharTx best on NER and MorphTx best on SA.

Original Arabic vs Transliterated Arabic Comparing the performance of adding Arabic data in different scripts we also observe similar trends. Except for Buckwalter, BERTu consistently attains better performance with transliterated data. In contrast, both mBERT and mBERTu perform worse with transliterated data, except for Uroman on SA.

In summary, data augmentation with some form of Arabic data generally improves performance. However, performance improvements with transliteration is dependent on the model's exposure to Maltese. Crucially, we highlight the importance of applying an appropriate transliteration scheme – while transliteration can eliminate script differences, its effectiveness relies on the orthographic similarities with the target language.

6 Cascaded Arabic Training

In Section 5, our results show that multilingual models do not benefit from cross-lingual transfer capabilities as much when fine-tuning with transliterated Arabic, instead of original Arabic. We hypothesise that this is due the model's pre-training on Arabic in Arabic script rather than Latin.

To test this, we conduct further experiments where we cascade multiple stages of fine-tuning on Arabic, with increasing similarity to Maltese, before the final stage of Maltese fine-tuning. Therefore, we first start with a fine-tuning step on original Arabic data, followed by transliterated Arabic data, and lastly a final phase of Maltese fine-tuning. To simplify the setup, we choose one transliteration system based on observed performance trends from Section 5 – MorphTx for NER and CharTx for SA. Furthermore, we consider another variant where we also fine-tune on machine translated Arabic after fine-tuning on original and transliterated Arabic.

We compare these cascaded approaches against fine-tuning with only one stage of Arabic fine-tuning – original, transliterated, and machine translated Arabic. The results are shown in Table 4.[3]

From our results, we observe that fine-tuning with MT is a competitive baseline. When fine-tuning with translations, better performance is obtained than all the previously presented approaches on NER with BERTu and on SA with mBERT.

With BERTu, cascaded Arabic fine-tuning does not help over a single phase of Arabic fine-tuning, as the best result is obtained with machine translation for NER and transliteration for SA. This is likely due BERTu's Maltese pre-training with little to no Arabic data. This is supported by the relatively lower scores observed for BERTu when fine-tuned with original Arabic.

In contrast, both mBERT and mBERTu achieve the best performance when fine-tuned on Arabic data in a cascaded approach. For mBERTu, the best result is obtained with original and transliteration cascading, whereas the full cascade yields the best result for mBERT. Interestingly, when mBERT and mBERTu are fine-tuned solely with either transliterated Arabic or translated Arabic, worse results are generally obtained compared to original Arabic fine-tuning, but by combining all approaches in one pipeline, we are better able to unlock the models' cross-lingual transfer capabilities.

[3] The numbers from Table 3 for Maltese only, original Arabic, and transliterated Arabic, are relisted to ease comparison.

	BERTu	mBERT	mBERTu
Maltese data only			
	58.3	50.1	64.9
Adding Original Arabic data			
	55.7	**60.8**	**69.5**
Adding Transliterated Arabic data			
Buckwalter	55.9	51.4	63.7
Uroman	56.5	53.2	64.8
CharTx	59.3	54.3	65.6
MorphTx	**61.3**	55.5	67.0

(a) Named-Entity Recognition (NER)

	BERTu	mBERT	mBERTu
Maltese data only			
	82.5	65.4	80.7
Adding Original Arabic data			
	82.3	**68.9**	79.4
Adding Transliterated Arabic data			
Buckwalter	80.8	65.6	77.2
Uroman	83.6	67.3	**81.4**
CharTx	**85.7**	67.0	78.8
MorphTx	82.4	65.1	78.0

(b) Sentiment Analysis (SA)

Table 3: Results from fine-tuning with transliterated Arabic, Arabic in the original script, and no Arabic data. The metrics used for NER and SA are span-level F1 and macro-averaged F1, respectively, and all scores are averages of 5 runs with different random seeds. Best scores per task and model are **bolded**.

Orig	Tx	MT	BERTu	mBERT	mBERTu
Maltese data only					
			58.3	50.1	64.9
Adding a single stage of Arabic fine-tuning					
✓			55.7	60.8	69.5
	✓		61.3	55.5	67.0
		✓	**67.3**	59.4	69.1
Adding multiple stages of Arabic fine-tuning					
✓	✓		58.6	61.6	**70.2**
✓	✓	✓	61.8	**61.8**	68.8

(a) Named-Entity Recognition (NER)

Orig	Tx	MT	BERTu	mBERT	mBERTu
Maltese data only					
			82.5	65.4	**80.7**
Adding a single stage of Arabic fine-tuning					
✓			82.3	68.9	79.4
	✓		**85.7**	67.0	78.8
		✓	82.5	69.3	78.7
Adding multiple stages of Arabic fine-tuning					
✓	✓		84.0	69.3	79.8
✓	✓	✓	82.7	**70.0**	76.7

(b) Sentiment Analysis (SA)

Table 4: Results from fine-tuning with different Arabic inputs: Original (Orig), Transliteration (Tx – MorphTx for NER and CharTx for SA), and Machine Translation (MT). The metrics used for NER and SA are span-level F1 and macro-averaged F1, respectively, and all scores are averages of 5 runs with different random seeds. Best scores per task and model are **bolded**.

7 Conclusion and Future Work

In this work, we presented transliteration systems from Arabic to Maltese. Our experimental results highlight that the effectiveness of transliteration depends on the model's exposure to the target language. We find that a monolingual Maltese model benefits from transliterated Arabic data, while multilingual models are able to make cross-lingual transfer links without transliteration. However, cascading the original and transliterated data during the fine-tuning process proves to be beneficial over fine-tuning with only one of these.

Future work includes exploring unsupervised or self-supervised methods to better align Arabic and Maltese representations without heavy reliance on parallel data or linguistic analysis. We also plan to investigate advanced machine translation and neural transliteration models that capture deeper morphological and phonological patterns. Additionally, we plan on applying these augmentation techniques to other facets of Maltese NLP such as language modelling.

8 Limitations

While our results demonstrate the potential of Arabic-driven augmentation for Maltese NLP, several limitations remain. First, the effectiveness of our approach is partly constrained by the quality of Arabic-to-Maltese translations, which may introduce stylistic or grammatical inconsistencies, especially when using off-the-shelf machine translation systems. Second, our transliteration rules and mappings, though linguistically motivated, simplify complex morphological and phonological relationships and may not generalise across all domains of Maltese usage. Third, our evaluations focus on downstream tasks using pre-existing datasets, which may not fully capture real-world variation in code-switching, informal registers, or dialectal usage. Finally, while we focus on Standard Maltese and Modern Standard Arabic, variation across dialects and registers in both languages is not addressed and may affect generalisability.

9 Ethical Considerations

Our study uses publicly available language resources and models for both Arabic and Maltese, adhering to the licencing and usage terms of each dataset. However, the use of machine translation for data augmentation carries the risk of reinforcing biases or introducing artefacts that may impact fairness and interpretability in downstream tasks. Moreover, while our approach aims to support a low-resource language, it assumes a certain level of equivalence between Arabic and Maltese that may obscure sociolinguistic or cultural distinctions. We encourage careful application of our methods, particularly in contexts involving sensitive or identity-related content. We also note that our transliteration and augmentation techniques are not intended for human communication and may not reflect idiomatic or culturally appropriate usage.

We used AI writing assistance within the scope of "Assistance purely with the language of the paper" described in the ACL Policy on Publication Ethics.

Acknowledgments

We thank Salam Khalifa and Rawan Bondok for their helpful feedback when preparing this paper. We also acknowledge funding by Malta Enterprise Research and Development Scheme.

References

Judit Ács. 2019. Exploring BERT's vocabulary. *Judit Ács's blog* (Accessed 2025-05-15).

Ramy Baly, Alaa Khaddaj, Hazem Hajj, Wassim El-Hajj, and Khaled Shaban. 2018. Arsentd-lev: A multi-topic corpus for target-based sentiment analysis in arabic levantine tweets. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Paris, France. European Language Resources Association (ELRA).

Yassine Benajiba, Paolo Rosso, and José Miguel Benedí Ruiz. 2007. ANERsys: An arabic named entity recognition system based on maximum entropy. In *Computational Linguistics and Intelligent Text Processing*, pages 143–153, Berlin, Heidelberg. Springer Berlin Heidelberg.

Tim Buckwalter. 2002. Buckwalter Arabic morphological analyzer version 1.0. Linguistic Data Consortium (LDC) catalog number LDC2002L49, ISBN 1-58563-257-0.

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.

Fadhl Eryani and Nizar Habash. 2021. Automatic Romanization of Arabic bibliographic records. In *Proceedings of the Sixth Arabic Natural Language Processing Workshop*, pages 213–218, Kyiv, Ukraine (Virtual). Association for Computational Linguistics.

Lucie Gianola, Ēriks Ajausks, Victoria Arranz, Chomicha Bendahman, Laurent Bié, Claudia Borg, Aleix Cerdà, Khalid Choukri, Montse Cuadros, Ona de Gibert, Hans Degroote, Elena Edelman, Thierry Etchegoyhen, Ángela Franco Torres, Mercedes García Hernandez, Aitor García Pablos, Albert Gatt, Cyril Grouin, Manuel Herranz, and 10 others. 2020. Automatic removal of identifying information in official eu languages for public administrations: The MAPA project. In *Proceedings of the 33rd International Conference on Legal Knowledge and Information Systems (JURIX'20)*, pages 223–226. IOS Press.

Nizar Habash, Fadhl Eryani, Salam Khalifa, Owen Rambow, Dana Abdulrahim, Alexander Erdmann, Reem Faraj, Wajdi Zaghouani, Houda Bouamor, Nasser Zalmout, Sara Hassan, Faisal Al-Shargi, Sakhar Alkhereyf, Basma Abdulkareem, Ramy Eskander, Mohammad Salameh, and Hind Saddiki. 2018. Unified guidelines and resources for Arabic dialect orthography. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Nizar Habash, Ramy Eskander, and Abdelati Hawwari. 2012. A morphological analyzer for Egyptian Arabic. In *Proceedings of the Twelfth Meeting of the Special Interest Group on Computational Morphology and Phonology*, pages 1–9, Montréal, Canada. Association for Computational Linguistics.

Nizar Y Habash. 2010. *Introduction to Arabic Natural Language Processing*. Morgan & Claypool Publishers.

Ulf Hermjakob, Jonathan May, and Kevin Knight. 2018. Out-of-the-box universal Romanization tool uroman. In *Proceedings of ACL 2018, System Demonstrations*, pages 13–18, Melbourne, Australia. Association for Computational Linguistics.

Go Inoue, Salam Khalifa, and Nizar Habash. 2022. Morphosyntactic tagging with pre-trained language models for Arabic and its dialects. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1708–1719, Dublin, Ireland. Association for Computational Linguistics.

Anne Lauscher, Vinit Ravishankar, Ivan Vulić, and Goran Glavaš. 2020. From zero to hero: On the limitations of zero-shot language transfer with multilingual Transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4483–4499, Online. Association for Computational Linguistics.

Yihong Liu, Chunlan Ma, Haotian Ye, and Hinrich Schütze. 2025. TransMI: A framework to create strong baselines from multilingual pretrained language models for transliterated data. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 469–495, Abu Dhabi, UAE. Association for Computational Linguistics.

Antonio Martínez-García, Toni Badia, and Jeremy Barnes. 2021. Evaluating morphological typology in zero-shot cross-lingual transfer. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3136–3153, Online. Association for Computational Linguistics.

Kurt Micallef, Fadhl Eryani, Nizar Habash, Houda Bouamor, and Claudia Borg. 2023. Exploring the impact of transliteration on NLP performance: Treating Maltese as an Arabic dialect. In *Proceedings of the Workshop on Computation and Written Language (CAWL 2023)*, pages 22–32, Toronto, Canada. Association for Computational Linguistics.

Kurt Micallef, Albert Gatt, Marc Tanti, Lonneke van der Plas, and Claudia Borg. 2022. Pre-training data quality and quantity for a low-resource language: New corpus and BERT models for Maltese. In *Proceedings of the Third Workshop on Deep Learning for Low-Resource Natural Language Processing*, pages 90–101, Hybrid. Association for Computational Linguistics.

Kurt Micallef, Nizar Habash, Claudia Borg, Fadhl Eryani, and Houda Bouamor. 2024. Cross-lingual transfer from related languages: Treating low-resource Maltese as multilingual code-switching. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1014–1025, St. Julian's, Malta. Association for Computational Linguistics.

Benjamin Muller, Antonios Anastasopoulos, Benoît Sagot, and Djamé Seddah. 2021. When being unseen from mBERT is just the beginning: Handling new languages with multilingual language models. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 448–462, Online. Association for Computational Linguistics.

Ossama Obeid, Nasser Zalmout, Salam Khalifa, Dima Taji, Mai Oudah, Bashar Alhafni, Go Inoue, Fadhl Eryani, Alexander Erdmann, and Nizar Habash. 2020. CAMeL tools: An open source python toolkit for Arabic natural language processing. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 7022–7032, Marseille, France. European Language Resources Association.

Fabian David Schmidt, Ivan Vulić, and Goran Glavaš. 2022. Don't stop fine-tuning: On training regimes for few-shot cross-lingual transfer with multilingual language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 10725–10742, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Genta Winata, Shijie Wu, Mayank Kulkarni, Thamar Solorio, and Daniel Preotiuc-Pietro. 2022. Cross-lingual few-shot learning on unseen languages. In *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 777–791, Online only. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, and 3 others. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Shijie Wu and Mark Dredze. 2019. Beto, bentz, becas: The surprising cross-lingual effectiveness of BERT. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 833–844, Hong Kong, China. Association for Computational Linguistics.

Shijie Wu and Mark Dredze. 2020. Are all languages created equal in multilingual BERT? In *Proceedings of the 5th Workshop on Representation Learning for NLP*, pages 120–130, Online. Association for Computational Linguistics.

Mengjie Zhao, Yi Zhu, Ehsan Shareghi, Ivan Vulić, Roi Reichart, Anna Korhonen, and Hinrich Schütze. 2021. A closer look at few-shot crosslingual transfer: The choice of shots matters. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5751–5767, Online. Association for Computational Linguistics.

A Transliteration System Rules

The character mappings presented in Section 3.1 are presented in Tables 5 and 6 while the morpheme mappings presented in Section 3.2 are shown in Table 7. We note how many of the character mappings include diacritics on the source side and are typically not used by the CharTx system (Section 3.1) since the text is not diacritised.

B Data Normalisation

Named-Entity Recognition (NER) Since the MAPA (Gianola et al., 2020) and ANERCorp (Benajiba et al., 2007) datasets have a different tagset, we normalise these to only keep Person (`PER`), Organisation (`ORG`), and Location (`LOC`) tags, dropping every other tag (`O`). For ANERCorp, this means that we only removed the Miscellaneous tags (`MISC`). However, the MAPA data has more fine-grained annotations. Hence, for MAPA, we only keep the `PER` the token's level 2 tag is either a `given name` or a `family name`, and we designate `LOC` for tokens marked as `city` or `country`. Originally, the MAPA data had 3,901 sentences which is comparable to the ANERCorp data. Since we are interested in data augmentation under a resource-constrained setting, we choose to downsample the MAPA training data to allow us to better measure this. Hence, we downsample the Maltese data, so that we have around the same ratio of sentences as the Arabic and Maltese datasets for Sentiment Analysis. By doing so, we end up with 155 and 43 sentences for training and validation, respectively, and the test set remains unchanged.

Sentiment Analysis (SA) The Maltese dataset from Martínez-García et al. (2021) only has positive or negative labels, whereas the Arabic dataset from Baly et al. (2018) has positive, negative, or neutral labels. Hence, we drop all Arabic sentences with a neutral label, ending up with 15,305 sentences. The Maltese data remains unchanged.

C Fine-Tuning Details

For our experimental setup described in Section 4, we fine-tune BERT-based models by adding a linear token or sentence classification head, depending on the task. We use the Transformers library to conduct all of our experiments (Wolf et al., 2020) and the code is made publicly available.[4]

For all tasks, we use an inverse square root learning rate schedule with a maximum learning rate of 2e-5 and a warmup of 1 epoch. We also set the classifier dropout to 0.1 and the weight decay to 0.01. We train with batch sizes of 16 for a maximum of 200 epochs early stopping on the development set[5] with a patience of 20 epochs. Each experiment is performed 5 times with different random seeds, reporting the average across these runs.

We fine-tune all models on a compute cluster using A100 GPUs. Fine-tuning runtimes vary largely because we train using early stopping but also depending on the dataset and model used. On average, a single training run on Arabic data takes around 28 minutes and 44 minutes for Named-Entity Recognition and Sentiment Analysis, respectively, while a single training run on Maltese data takes around 7 minutes and 3 minutes for Named-Entity Recognition and Sentiment Analysis, respectively.

[4] `https://github.com/MLRS/BERTu/tree/main/finetune`

[5] The development set used is always the Maltese data, even when fine-tuning with Arabic data.

Source	Target
ل *l* + Sun Letter with Gemination	
لدّ *ld~* / لذّ *l*~* / لضّ *lD~* / لظّ *lZ~*	dd
للّ *ll~*	ll
لنّ *ln~*	nn
لرّ *lr~*	rr
لسّ *ls~* / لصّ *lS~*	ss
لتّ *lt~* / لثّ *lv~* / لطّ *lT~*	tt
لشّ *l$~*	xx
لزّ *lz~*	żż
Final ي *y* with Gemination	
`[EOS]`يّ *y~*	i
Gemination	
بّ *b~*	bb
دّ *d~* / ذّ **~* / ضّ *D~* / ظّ *Z~*	dd
فّ *f~*	ff
هّ *h~*	hh
جّ *j~*	ġġ
حّ *H~* / خّ *x~*	ħħ
عّ *E~* / غّ *g~*	għ
يّ *y~*	jj
كّ *k~*	kk
لّ *l~*	ll
مّ *m~*	mm
نّ *n~*	nn
قّ *q~*	qq
رّ *r~*	rr
سّ *s~* / صّ *S~*	ss
تّ *t~* / ثّ *v~* / طّ *T~*	tt
وّ *w~*	ww
شّ *$~*	xx
زّ *z~*	żż

Source	Target
Hamza/Alif with Diacritic	
ءْ *’o* / اً *AF* / اٍ *AK* / اٌ *AN*	
ءَ *’a* / أَ *>a* / ٱَ *{a* / آَ *|a*	a
ءِ *’i* / إِ *<i* / ٱِ *{i* / آِ *|i*	i
ءُ *’u* / أُ *>u* / ٱُ *{u* / آُ *|u*	u
Long Vowel ‘*a*’	
◌َا *aA*	a
◌َى *aY*	a
◌َة *ap*	a
‘*i*’/‘*y*’ Glide	
يا *yA*	ja
يْ *yo*	j
يَ *ya* / ئَ *}a*	je
يِ *yi* / ئِ *}i*	ji
يُ *yu* / ئُ *}u*	ju
◌ْي *oy*	j
◌َي *ay*	ej
◌ُي *uy*	uj
◌ِي *iy*	i
ئْ *}o*	i
‘*u*’/‘*w*’ Glide	
وا *wA*	wa
وْ *wo*	w
وَ *wa* / ؤَ *&a*	we
وِ *wi* / ؤِ *&i*	wi
وُ *wu* / ؤُ *&u*	wu
◌ْو *ow*	w
◌َو *aw*	ew
◌ِو *iw*	iw
◌ُو *uw*	u
ؤْ *&o*	u

Table 5: Character Mappings (multiple characters) outlining how source Arabic characters are mapped to target Maltese characters. The Buckwalter representation for Arabic is also shown. `[BOS]` and `[EOS]` are special markers indicating the beginning and end positions of a word.

Source	Target
Diacritics	
◌ٰ `	a
◌َ *a*	e
◌ِ *i*	i
◌ُ *u*	u
◌ْ *o* / ◌ً *F* / ◌ٍ *K* / ◌ٌ *N* / ◌ّ ~	
Special Characters at Word Boundaries	
`[EOS]`ع *E* / `[EOS]`غ *g*	’
أ`[BOS]` > / إ`[BOS]` <	
Letters	
ا *A* / أ > / آ | / ى *Y*	a
ي *y* / إ < / ئ }	i
و *w* / ؤ &	u
ء ’ / ٱ {	
ة *p*	a
ب *b*	b
د *d* / ذ * / ض *D* / ظ *Z*	d
ف *f*	f
ج *j*	ġ
ه *h*	h
ح *H* / خ *x*	ħ
ع *E* / غ *g*	għ
ك *k*	k
ل *l*	l
م *m*	m
ن *n*	n
ق *q*	q
ر *r*	r
س *s* / ص *S*	s
ت *t* / ث *v* / ط *T*	t
ش $	x
ز *z*	ż

Source	Target
Symbols	
،	,
؛	;
؟	?
٪	%
٩	9
٨	8
٧	7
٦	6
٥	5
٤	4
٣	3
٢	2
١	1
٠	0

Table 6: Character Mappings (single characters) outlining how source Arabic characters are mapped to target Maltese characters. The Buckwalter representation for Arabic is also shown. `[BOS]` and `[EOS]` are special markers indicating the beginning and end positions of a word.

Tag	Source	Target
CONJ	وَ *wa* / وِ *wi*	u_
DET	اَل *Al*	il-
PREP	بِ *bi*	bi_
PREP	لِ *li*	li_
PREP	فِي *fiy*	fi_
NOUN	مَع *maE*	ma’_
NOUN	تَاع *taAE*	ta’_
PREP	عَلَى *EalaY*	għal_
PREP	مِن *min*	minn_
NSUFF_FEM_SG (construct state)	ة *p*	t
FUT_PART	سَ *sa*	sa
CASE_*_*	*	
IVSUFF_MOOD:*	*	
IVSUFF_SUBJ:2FS	*	
IVSUFF_SUBJ:{D,MP,FP}	*	u
PVSUFF_SUBJ:{1S,2MS,2FS}	*	t
PVSUFF_SUBJ:3MS	*	
PVSUFF_SUBJ:3FS	*	at
PVSUFF_SUBJ:1P	*	na
PVSUFF_SUBJ:{2D,2MP,2FP}	*	tu
PVSUFF_SUBJ:3MP	وْا *woA*	ew
PVSUFF_SUBJ:{3MD,3FD,3MP,3FP}	*	u
CVSUFF_SUBJ:{2MS,2FS}	*	
CVSUFF_SUBJ:2MP	*	u
PRON_1S	نِي *niy*	ni
{PRON,POSS_PRON}_1S	*	i
{PRON,POSS_PRON}_{2MS,2FS}	*	ek
PRON_3MS	*	u
POSS_PRON_3MS	*	u
{PRON,POSS_PRON}_3FS	*	ha
{PRON,POSS_PRON}_1P	*	na
{PRON,POSS_PRON}_{2D,2MP,2FP}	*	kom
{PRON,POSS_PRON}_{3D,3MP,3FP}	*	hom
NSUFF_FEM_SG	*	a
NSUFF_MASC_DU_*	*	ejn
NSUFF_FEM_DU_*	*	tejn
NSUFF_MASC_PL_*	*	in
NSUFF_FEM_PL	*	iet

Table 7: Morpheme Mappings indicating how different morpheme classes are mapped; some tags are collapsed to a single row for presentation purposes. When the Arabic form is specified in the Source column, the target mapping is only applied to this specific form, otherwise it applies to all forms (indicated by *). ‘_’ indicates an explicit spacing added so that the morpheme is no longer attached to the word.