Multi-Teacher Distillation for Pretraining a Low-Resource Language

Amit Kumar Chaudhary

MSc. Dissertation

Department of Artificial Intelligence
Institute of Linguistics and Language Technology
Faculty of Information and Communication Technology

University of Malta

October, 2023

Supervisor(s):
Dr. Claudia Borg, Department of Artificial Intelligence
Kurt Micallef, Department of Artificial Intelligence
Prof. Roberto Zamparelli, University of Trento

Submitted in partial fulfilment of the requirements for the degree of
Master of Science in Human Language Science and Technology (HLST)

Supervisor(s)

Dr. Claudia Borg

Department of Artificial Intelligence

University of Malta

and

Prof. Roberto Zamparelli

Department of Psychology and Cognitive Science

University of Trento

Co-supervisor(s)

Kurt Micallef

Department of Artificial Intelligence

University of Malta

Acknowledgements

First of all, I am grateful to my supervisors Dr. Claudia Borg from the University of Malta and Prof. Roberto Zamparelli from the University of Trento for their valuable feedback and guidance during this thesis work.

I would also like to thank my co-supervisor Kurt Micallef at the University of Malta, for his assistance in addressing technical issues, offering suggestions on implementation details, and providing guidance when I encountered challenges.

I am also thankful to the LT-Bridge Project (GA 952194) and the Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI) for their support in providing the Virtual Laboratory, which was used to train the deep learning models used in this thesis.

I'm indebted to the Language and Communication Technologies (LCT) master's program for awarding me a full scholarship that allowed me to pursue a two-year degree in Europe, laying the foundation for this thesis work.

I would also like to extend my appreciation to my friends from the LCT program and those in local programs who shared the journey of the degree with me over the past two years in Italy and Malta.

Finally, I would like to acknowledge my friends and family back home in Nepal for their moral support during my abroad study.

Abstract

Masked Language Models like BERT have brought a paradigm shift in the field of Natural Language Processing, achieving state-of-the-art results across numerous NLP tasks. However, pretraining such models for low-resource languages remains a considerable challenge. Existing approaches involve either training language-specific BERT models by collecting large text corpora or using massively multilingual models, which, unfortunately, face the challenge of per-language performance drop when handling multiple languages in a single model termed as "curse of multilinguality". A promising solution came in the form of MergeDistill (Khanuja et al., 2021), which can combine monolingual models from similar languages into a unified multilingual model using Knowledge Distillation and finds that the combined model performs better than each individual model. However, this approach remains unexplored for low-resource languages.

Our research aims to fill this gap by adapting MergeDistill for the low-resource language Maltese. We aim to understand the impact of pretraining masked language models for such low-resource languages through the method of knowledge distillation from high-resource languages that are linguistically related. Our evaluation focuses on the Maltese language, involving knowledge distillation from a monolingual Maltese teacher model, as well as teachers from Arabic, English, and Italian. We analyze the performance differences among monolingual, massively multilingual, and language-specific multilingual models on both semantic and syntactic tasks. Our results indicate that knowledge distillation enhances the efficiency of pretraining, and the inclusion of related languages contributes positively to the evaluation tasks. Furthermore, these combined models retain most of their performance in evaluation tasks, despite encountering fewer pretraining data and being trained for less training time than monolingual models.

Contents

1 Introduction

The release of BERT (Devlin et al., 2018), an encoder-only variant of the transformer model in 2018 was a significant development in the field of NLP. Prior work used to focus on developing specialized architectures manually crafted for a specific NLP task and advancements in one task wouldn't benefit others. BERT, on the other hand, brought two major paradigm changes. First was the concept of unsupervised pretraining where a general-purpose model learns contextual representations in a completely unsupervised manner from large text corpus such as Wikipedia. The training included tasks like Masked Language Modeling, where the goal is to guess a hidden word from the other words, and Next Sentence Prediction, where the model decides if one sentence follows another. The second paradigm change was the use of the same model architecture across many NLP tasks with minimal modification. For instance, after training the BERT model, you can fine-tune it for various text-related tasks, like sentiment analysis, understanding relationships between texts (entailment), answering questions, rewriting text (paraphrasing), measuring similarity between texts, and more. Fine-tuning pre-trained models also lowered the required data amount for achieving good performance. When released, BERT got state-of-the-art results in the English GLUE(Wang et al., 2018) benchmark.

However, bringing advancements like BERT from English to low-resource languages poses significant challenges. Pre-training these models requires a large high-quality text corpus to learn the tokenizers and model representations. However, many low-resource languages lack the availability of enough text corpus online. For comparison, English Wikipedia has 6.6 million articles while Maltese Wikipedia has only 5172 articles [1]. Training on such a small text corpus can lead to overfitting and prevent learning general-purpose representations.

Multilingual BERT models were proposed as one solution where the BERT style pretraining was applied to a corpus composed of multiple languages to learn shared representations. The idea was that pretraining on high-resource languages would benefit related low-resource languages. However, Conneau et al. (2020) show that these multilingual

[1] `https://meta.wikimedia.org/wiki/List_of_Wikipedias`

models suffer from the curse of multilinguality where the performance of each language suffers when the model is expanded to more languages due to a fixed model capacity. Evaluation of such models has also shown that while the performance on high-resource languages is good, these multilingual models perform worse than the baseline on low-resource languages (Wu and Dredze, 2020). These multilingual models also only support a fixed set of languages, with the highest resources, and as such, many languages are left behind. As an example, mBERT(Devlin et al., 2019) focuses on the 104 languages with the most Wikipedia articles.

Maltese is an example of a low-resource language (Rosner and Borg, 2022) excluded in mBERT. It is spoken in Malta and is particularly interesting in our study because it's the only Semitic language that uses the Latin script (Brincat, 2011). The grammatical structure of Maltese is influenced by Semitic languages, while its vocabulary is heavily borrowed from English and Romance languages like Italian. Micallef et al. (2022) tackles this lack of large text corpora by extending existing corpora of Maltese text called Korpus Malti with texts from more data sources. They pre-train a monolingual BERT model on Korpus Malti called BERTu as well as further pretraining mBERT to include Maltese to create a multilingual model called mBERTu. Their research showed that models trained on Korpus Malti outperform those trained solely on Wikipedia. Additionally, the monolingual model performed better than the multilingual model in three out of four evaluation tasks. However, the monolingual model misses out on the cross-lingual capabilities found in the multilingual model.

Knowledge Distillation is another promising direction in the field for data efficiency. Knowledge Distillation was originally proposed in the domain of Computer Vision for a digit classification task(Hinton et al., 2015). The concept involves teaching a smaller student model to imitate the predictions of a bigger teacher model. In typical supervised training, the model learns by comparing its predictions to the actual label, which is typically a probability distribution where one category is assigned a probability of 1 (correct category) and others have a probability of 0. However, predictions for a trained model are probabilities over the classes. The probabilities besides the predicted class also provide useful information. For example, assume a model doing image classification gives

a 98% probability for the image to be a bicycle and the category with the second highest probability could be a motorbike. This makes intuitive sense as both share semantic properties like being vehicles, having similar shapes, etc. Thus, the output of already trained models has higher information compared to a ground-truth label. Knowledge Distillation was originally seen as a model compression approach since the student model can be smaller than the teacher and thus applicable when deploying these models in resource-crunched environments. However, it also provides benefits as a data-efficient training method as the student model can learn much faster when distilled from the teacher's predictions compared to training only from scratch.

The idea of knowledge distillation is also applicable in pretraining BERT-based models. Given that the objective of BERT is to predict the masked tokens, a trained BERT model would give probabilities over the vocabulary for each token as seen in Figure 1. The probabilities for a `[MASK]` token are valuable as there can be multiple words that fit the context. This idea has been leveraged to distill and compress large BERT-based models acting as teacher to smaller student BERT models (Sanh et al., 2019; Sun et al., 2019; Jiao et al., 2020; Sun et al., 2020) with a few percentage drop in performance.

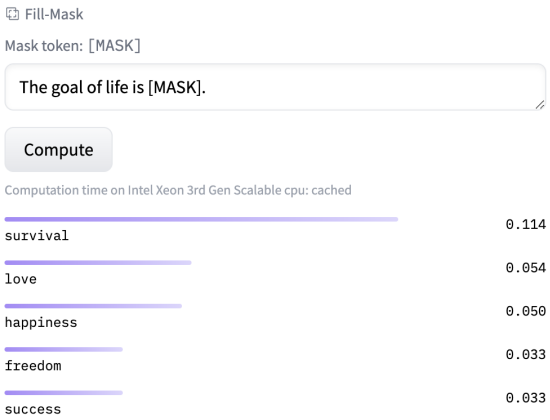

Figure 1: Predictions from BERT for a `[MASK]` token

MergeDistill (Khanuja et al., 2021) apply Knowledge distillation in the context of BERT pretraining by combining monolingual models from similar languages into a unified multilingual model using Knowledge Distillation and found that the combined model performs better than each monolingual model. This is an interesting approach as this

brings us the benefits of multilinguality while not losing the performance advantages of monolingual models.

However, the MergeDistill approach has not been explored for low-resource languages. In our work, we tackle this research gap in the context of Maltese. Given that languages related to Maltese – Arabic, English, and Italian all have strong pre-trained BERT-based models available, we use the MergeDistill framework to pretrain a multilingual Maltese BERT model in a low resource scenario leveraging only the small Wikipedia corpus for Maltese.

1.1 Overview of the work carried out

In this section, we'll give you an overview of what we did to achieve the goals outlined in this thesis.

- We replicate the MergeDistill framework, which has no prior public code implementation yet, independently using the latest Huggingface ecosystem of transformers and datasets libraries. We keep our implementation language-independent for easy extensibility to other low-resource languages

- We generate pre-processed versions of the Wikipedia corpus for English, Italian, Maltese, and Arabic as well as provide pretraining corpora for masked language modeling annotated with predictions from the respective teacher BERT-based models suitable for distillation

- We validate knowledge distillation in a single language setting by distilling a monolingual Maltese BERT model into a smaller student model using only the Maltese Wikipedia and evaluate it against a model trained from scratch on the downstream tasks

- We also evaluate a multi-teacher distillation setting by merging a monolingual Maltese model BERTu with teacher models from three related languages – Italian, Arabic, and English and evaluating it against a single-teacher distillation only from BERTu

- We compare this specific multilingual model composed of only related languages merged via distillation to monolingual and massively multilingual models on various downstream tasks for Maltese

1.2 Thesis Structure

Chapter 2 describes our main research question and how we aim to answer it in this thesis.

Chapter 3 describes the background theory on pre-training of BERT-based models, subword tokenization, and the adaptation of this pre-training in a multilingual context, and concludes with a brief overview of knowledge distillation for transformer models and the huggingface ecosystem of libraries.

Chapter 4 introduces the phenomenon of the curse of multilinguality and outlines the development of language-specific BERT models for different languages. We proceed to describe the MergeDistill framework, which serves as a middle ground between the two approaches and finally tie these discussions back to their relevance in achieving the goal of this thesis.

Chapter 5 presents our experimental framework on how we adapted the MergeDistill framework for Maltese.

Chapter 6 presents the evaluation criteria and describes both the semantic and sequence labeling downstream tasks. We also describe how we modify the base architecture for these tasks and the associated metrics used for each task.

Chapter 7 presents the results from our experiments. We compare Knowledge Distillation to supervised training on Maltese Wikipedia, run a comparison of the distillation using multiple teachers against baselines, and compare those models against monolingual and massively multilingual models.

Chapter 8 provides a summary of the findings and results stemming from our experiments, in addition to outlining possible areas for future research.

2 Aims and Objectives

In this thesis, we aim to answer the general research question "What is the comparative performance of specific multilingual models composed of only related languages against monolingual and massively multilingual models for low-resource languages?".

We address this research question for the low-resource language Maltese by conducting experiments with the following objectives:

1. Evaluate the effectiveness of Knowledge Distillation to pretrain a student model by distilling from an existing monolingual BERT-based model for Maltese when provided a limited amount of text corpora

2. Implement the MergeDistill (Khanuja et al., 2021) framework to merge the Maltese monolingual model with each of the monolingual models from Arabic, Italian, and English and evaluate it on downstream tasks

3. Compare the performance difference between distilling from a single Maltese monolingual model vs distilling from multiple teacher models

4. Compare the performance of the specific multilingual BERT model obtained via distillation to that of existing monolingual and multilingual model for Maltese using downstream tasks

3 Background

In this chapter, we describe the background knowledge relevant to our work. Section 3.1 describes the pre-training details of masked language models like BERT and we go over how these models make use of subword tokenization in Section 3.2. We describe how BERT-based models have been extended to train on multiple languages in Section 3.3. Furthermore, we introduce the concept of Knowledge Distillation and how it has been adapted to various NLP tasks in Section 3.4. Finally, we conclude with a description of various libraries in the Huggingface Ecosystem that we use for our work in Section 3.5.

3.1 Masked Language Models

Devlin et al. (2018) proposed a transformer-based encoder model called BERT that could learn representations from unlabeled text by being trained to predict randomly masked tokens (Masked Language Modeling). BERT achieved state-of-the-art results on the GLUE benchmark (Wang et al., 2018) for English. This has made transfer learning mainstream by providing a single architecture that could be easily fine-tuned for different types of NLP tasks.

In BERT, they use a pretraining objective called Masked Language Modeling (MLM). In MLM, random tokens in a sentence are hidden, and the model's task is to guess what these hidden tokens are, as shown in Figure 2. Specifically, for MLM, they mask 15% of the tokens in the input. Among these 15%, 80% are replaced with a special token `[MASK]`, 10% are swapped with random words from the vocabulary, and the remaining 10% are left unchanged. This was done because during pretraining, the BERT model would be exposed to `[MASK]` tokens while during finetuning and in actual downstream applications, it would never be fed `[MASK]` token but the entire sequence as is.

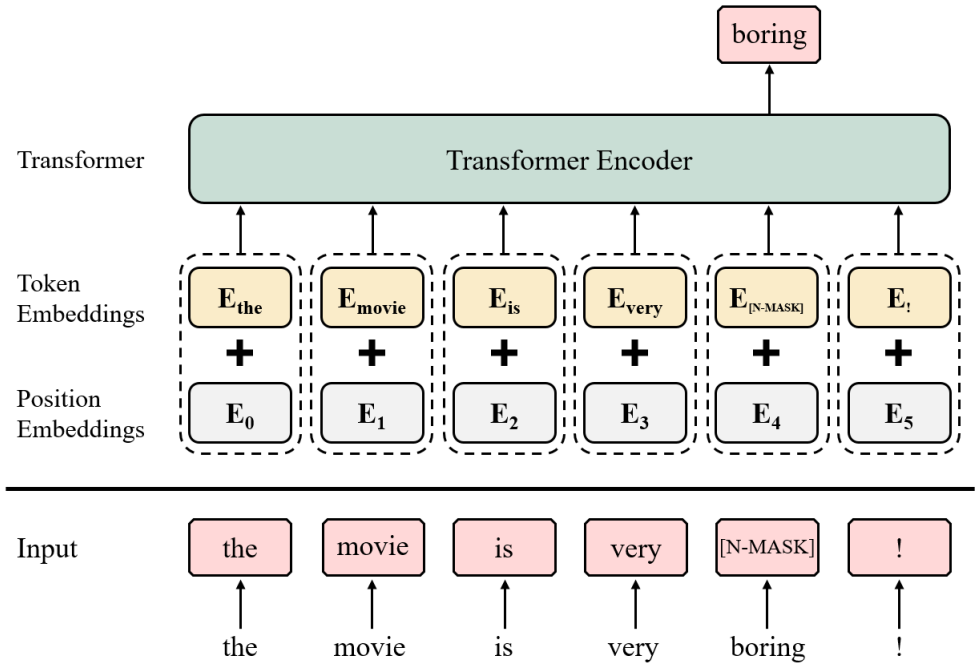

Figure 2: Illustration of Masked Language Modeling Objective in BERT (Park and Ahn (2019))

This pretraining objective doesn't rely on human-annotated data; instead, it can be trained using a vast amount of unlabeled text readily available on the internet. BERT was pre-trained on 16GB of unlabeled raw text extracted from the English Wikipedia as well as the BookCorpus (Zhu et al., 2015).

For pretraining, they use a BERT architecture that accepts a maximum input sequence of 512 tokens. Since the attention mechanism scales with input sequence length, they pretrain the BERT model in two phases to speed up training. For 90% of the training, they generated input sequences of 128 tokens, and then the remaining 10% of the training used 512 tokens. The model was pre-trained for a total of one million steps.

BERT also made use of an additional objective called Next Sentence Prediction. In this, the model is given sets of sentences and must guess if the second sentence can come after the first one. This is also unsupervised as we can generate positive and negative examples automatically from raw text corpus without any human input. The Next Sentence Prediction task was suggested with the intuition that it would improve performance on tasks that require a pair of texts.

To see how good BERT is compared to older methods, the author tests the model on different tasks in the GLUE benchmark. This includes two sentence entailment tasks. Multi-Genre Natural Language Inference(MNLI) is a multiclass entailment task to determine if a new sentence either supports, opposes, or has no clear relationship with another sentence (Williams et al., 2018). Recognizing Textual Entailment(RTE) is another entailment task but it only has two classes (Wang et al., 2018). Another task related to entailment in GLUE is the Question Natural Language Inference (QNLI) task (Wang et al., 2018). It's about deciding whether a sentence can answer a question. The Quora Question Pairs (QQP)[2] task is all about deciding if two questions from Quora are saying the same thing (paraphrases) or not. MRPC (Dolan and Brockett, 2005) and STS-B (Cer et al., 2017) instead of comparing questions focus on checking whether sentences are paraphrases of each other. Corpus of Linguistic Acceptability(CoLA) is a task to check grammatical correctness of a sentence (Warstadt et al., 2019). The Semantic Textual Similarity Benchmark(STS-B) task is about figuring out how much two sentences resemble each other, and this is rated on a scale from 1 to 5(Cer et al., 2017). The Stanford Sentiment Treebank (SST-2) is a task where you classify the sentiment of a text into positive or negative (Socher et al., 2013).

One significant benefit of BERT is that it can be readily adjusted for various tasks without the need for substantial changes to its architecture. As seen in Figure 3(a) & (b), for a single sentence or a pair of sentence classification, you can simply feed the output of the final layer for the `[CLS]` token to a feedforward layer to predict the classes. For a token-classification task such as NER or POS tagging as seen in Figure 3(d), you can use the output embeddings for each token in the final layer to a feedforward layer to predict the classes.

Follow-up work on BERT-based models uses a larger amount of pretraining data. Liu et al. (2019) scaled the pretraining text corpora from the original 16GB used in BERT to 160GB by leveraging additional texts from Reddit (Gokaslan and Cohen, 2019) and Common Crawl News (Nagel, 2016) in addition to Wikipedia. They also discovered that the Next Sentence Prediction task didn't offer as much benefit for other tasks. As a result,

[2] `https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs`

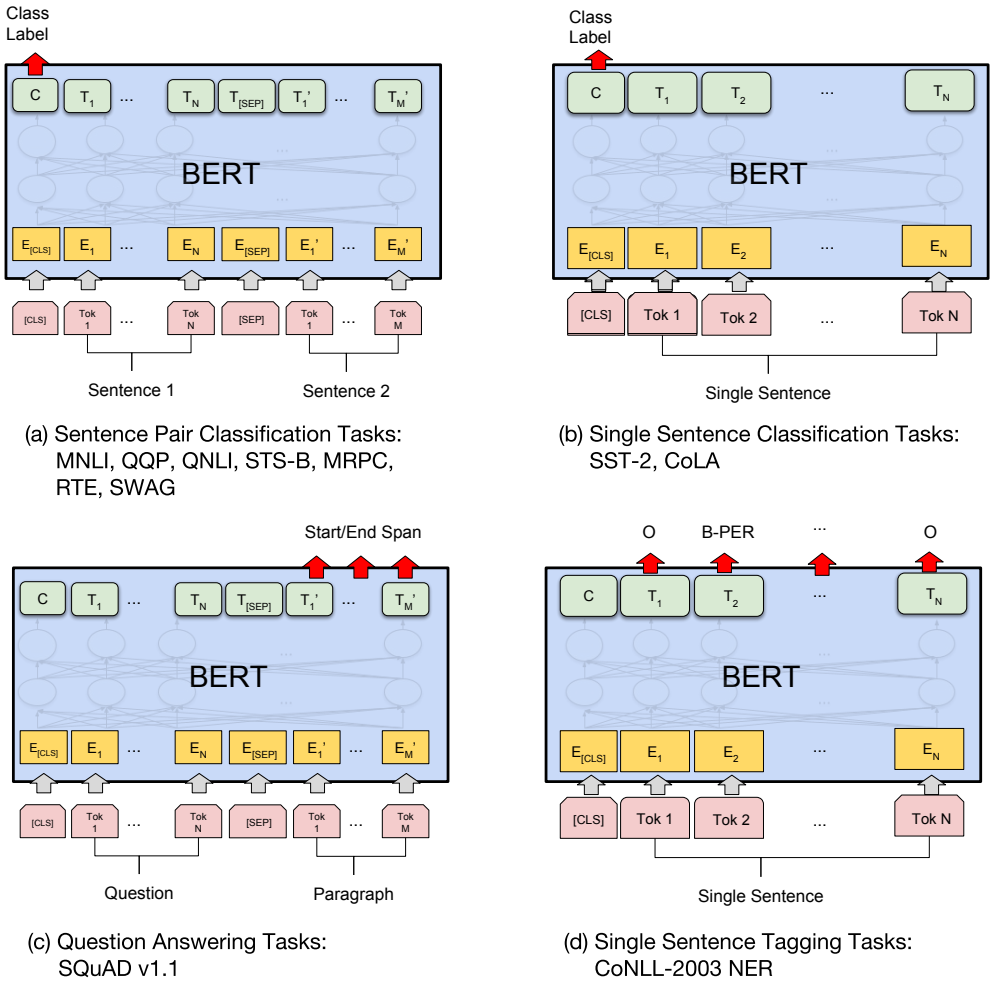

Figure 3: Adapting BERT to different downstream tasks. Figure reprinted from Devlin et al. (2018)

they developed a model similar to BERT, named RoBERTa, by training it exclusively with the Masked Language Modeling objective.

3.2 Subword Tokenization

Conventionally, when it comes to tokenization in natural language processing, two main approaches have been utilized. The first approach involves the use of rule-based tokenizers (Penn Treebank Tokenizer [3], Spacy (Honnibal et al., 2020), etc.), which are designed

[3] `https://www.nltk.org/_modules/nltk/tokenize/treebank.html`

based on linguistic rules specific to each language. They break the input text into tokens separated by spaces, punctuation marks, and periods and handle special cases like contractions and so on. These tokenizers, although effective, require extensive rule development and maintenance for each supported language. Also, models need to map these tokens to numeric indices for embeddings, and as vocabulary size has to be limited, there is an issue of out-of-vocabulary (OOV) tokens when some token not seen during training is present in the input text.

The second approach is character-level tokenization, where the text is broken down into individual characters. While this can be language-independent and also solve the out-of-vocabulary token issue, it can generate very long sequences. Also, a character may not adequately capture the meaningful subunits of words in different languages.

Subword tokenization offers a compromise between these two methods. Rather than considering every word as one token, subword tokenization divides words into smaller components, such as syllables or combinations of letters. For example, if the word 'cats' is not in our vocabulary but 'cat' and 's' are, then a new word 'cats' will be broken down into 'cat' and 's'. This is particularly useful for languages with rich morphology and handling out-of-vocabulary words.

BERT utilizes a subword tokenization method known as WordPiece, as described in Schuster and Nakajima (2012) and Wu et al. (2016). This tokenization process is learned automatically from a substantial amount of text in the target language, rather than relying on manually created rules.

Let's take a simple example to understand how it works. Suppose our entire text corpus was just the sentence "cat cat cat cat cat dog dog bat bat bat". WordPiece initially separates the text by breaking it at spaces and punctuation marks. So, we would get

Corpus: `("cat", 5), ("dog", 2), ("bat", 3)`

Now, for each word, we create an initial vocabulary by splitting each word into the first character as it is and the remaining characters preceded by `##`.

Corpus: `("c" "##a" "##t", 5), ("d" "##o" "##g", 2), ("b" "##a" "##t",`

3)

Vocab: `["c", "##a", "##t", "d", "##o" "##g", "b"]`

Now, we merge the pair that has the highest score among all the pairs. The score for a pair x and y is defined below

$$score(x, y) = \frac{\text{frequency(x, y)}}{\text{frequency(x)} * \text{frequency(y)}}$$

In our example, we can calculate the score for "##o" and "##g" as:

$$score(\texttt{"##o"}, \texttt{"##g"}) = \frac{\texttt{freq("##o", "##g")}}{\texttt{freq("##o")} * \texttt{freq("##g")}} = \frac{2}{2 * 2} = 0.5$$

$$score(\texttt{"##a"}, \texttt{"##t"}) = \frac{\texttt{freq("##a", "##t")}}{\texttt{freq("##a")} * \texttt{freq("##t")}} = \frac{8}{8 * 8} = 0.125$$

So, we merge the `"##o"` and `"##g"` into `"##og"` and add it to vocabulary and update the corpus.

Corpus: `("c" "##a" "##t", 5), ("d" "##og", 2), ("b" "##a" "##t", 3)`

Vocab: `["c", "##a", "##t", "d", "##o" "##g", "b", "##og"]`

Next we merge the `"d"` with `"##og"` and add `"dog"` to vocabulary.

Corpus: `("c" "##a" "##t", 5), ("dog", 2), ("b" "##a" "##t", 3)`

Vocab: `["c", "##a", "##t", "d", "##o" "##g", "b", "##og", "dog"]`

Assuming we had a target vocabulary size of 9, we stopped going further. Once our vocabulary is learned, we can now apply the tokenizer.

If we receive an input text "dog", we tokenize it directly as `["dog"]` since that's the longest match from the start available in the vocabulary. A text like "cat" would be tokenized as `["c", "##a", "##t"]`. We can also tokenize new words not present in a corpus such as `cog` into `["c", "##og"]` if the subwords are present in the vocabulary. If we receive a word that includes a character not present in our vocabulary such as "dogs", it would get tokenized into `["dog", "[UNK]"]`. As such, even if a part of the word exists in the vocabulary, if there is any unknown character, the whole word is mapped to a special unknown token `[UNK]`.

The WordPiece tokenizer in bert-base-cased learns a vocabulary size of 28,996 using the same procedure. It also includes special tokens such as `[SEP]` to indicate the end of the sentence and a separator for text pairs and `[CLS]` to act as a classification token added to the beginning. Since it's trained on 16GB of text, it would be rare to encounter `[UNK]` tokens as all single-letter characters would be present in the vocabulary. In the worst case, the whole word would be tokenized into single characters.

3.3 Massively Multilingual Models

While BERT (Devlin et al., 2018) was only trained in English, there have been efforts to train a BERT-based model that can handle multiple languages at once.

Devlin et al. (2019) proposed a multilingual variant of BERT called mBERT by using the same underlying architecture and pretraining procedure as BERT but jointly training on 104 languages. They concatenate the entire Wikipedia dump for the 104 languages and learn a shared vocabulary of size 110K for the WordPiece tokenizer using the procedure described in Section 3.2. However, learning a tokenizer and representations directly on this corpus would lead to low-resource languages being underrepresented. Thus, exponential smoothing with a value of 0.7 is applied to undersample high-resource languages and oversample low-resource languages to create a balanced corpus, and then the generation of WordPiece vocabulary and pretraining is done on this.

Lample and Conneau (2019) on the other hand, expand upon the standard masked language modeling (MLM) objective by introducing a new objective known as Translation Language Modeling (TLM). As seen in Figure 5, the MLM objective is similar to mBERT except that they also add an additional language embedding to all the input tokens to indicate the language explicitly. The Translation Language Modeling(TLM) objective is a supervised objective where parallel sentences from two languages that are translations of each other are masked and fed to the model at once. The idea behind this was that it would enable the model to connect representations from two languages because context from one language could be used to predict masked tokens in the other. XLM outperformed mBERT on the XNLI benchmark (Conneau et al., 2018) for zero-shot cross-lingual classification. In this scenario, the model, which was only fine-tuned on English data, was evaluated

across 15 different languages in a zero-shot manner.

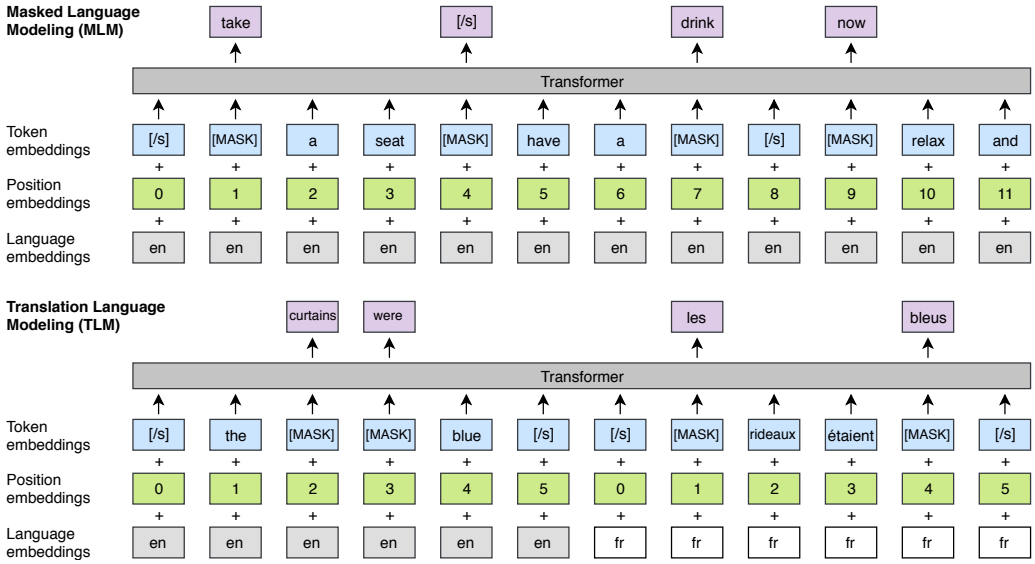

Figure 4: Illustration of two pretraining objectives in XLM. Figure reprinted from Lample and Conneau (2019)

Conneau et al. (2020) use the RoBERTa playbook of scaling up the dataset size and the training duration in their multilingual model called XLM-R. It is trained using the MLM objective on 100 languages. Instead of using Wikipedia as done in mBERT, they instead scale up the dataset size to 2.5TB by using a web-crawled corpus to get monolingual data for each language from CommonCrawl (Wenzek et al., 2019). They also remove the Translation Language Modeling (TLM) objective as that requires supervised parallel data and is a bottleneck when scaling up to 100 languages. Text corpora scraped from the web are going to be noisy and exhibit code-mixing of languages. As such, the author removed the language embeddings used in XLM-R which indicate the language code for the inputs. They also use a larger shared vocabulary size of 250k tokens in their BPE tokenizer (Sennrich et al., 2016).

3.4 Knowledge Distillation

Knowledge Distillation is a method, as described by Bucila et al. (2006); Hinton et al. (2015), to make a model smaller in size without losing much performance. This is achieved

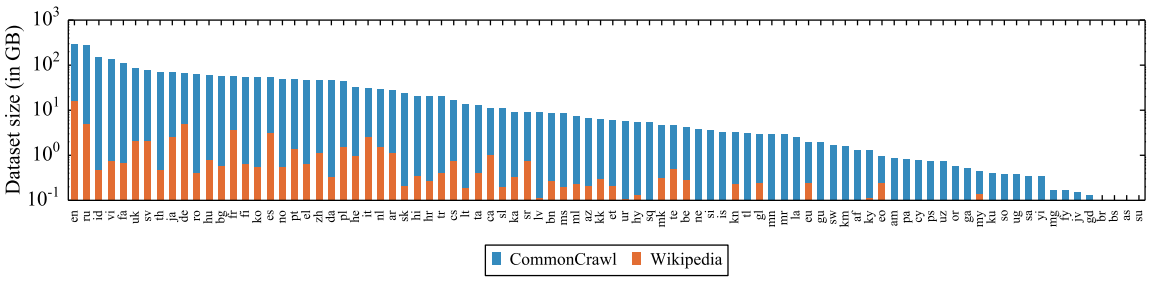

Figure 5: Comparison of Wikipedia vs CommonCrawl for 100 languages. Figure reprinted from Conneau et al. (2020)

by training a smaller neural network to imitate the output of a larger teacher network. The student network learns from both the actual labels and the teacher's predictions.

In a typical supervised model, the model makes predictions for the classes, and its output is compared to a one-hot-encoded ground truth label using cross-entropy loss. In knowledge distillation, in addition to the hard labels, the model also learns to predict the softer, more nuanced targets provided by a pre-trained teacher model, as illustrated in Figure 6.

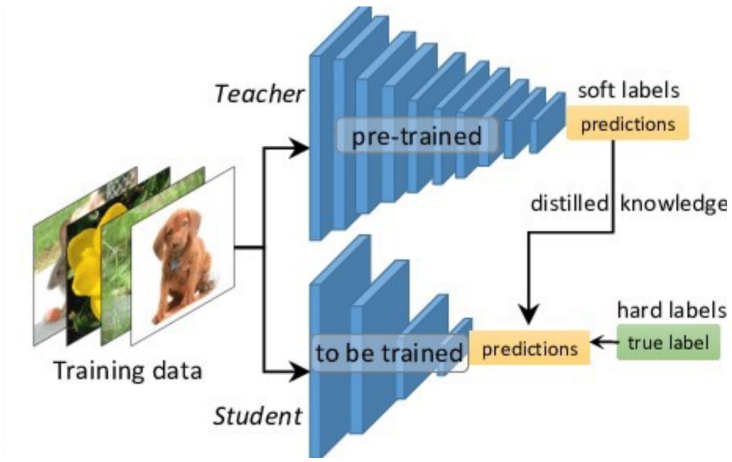

Figure 6: Illustration of Knowledge Distillation in Image Classification. Figure reprinted from Ganta et al. (2021)

Hinton et al. (2015) illustrate this in a digit classification task called MNIST and found a small distilled model to have a lower test error than a model trained only on hard labels.

In addition to using a single teacher, Hinton et al. (2015) also shows an approach of distilling an ensemble of teacher models trained on the same task into a single student model. This is done by averaging the output probability from the different models over the classes and having the student model match this averaged distribution. The advantage of the distillation approach is that the student architecture and training dataset can be different from the one used to train the teacher.

Knowledge Distillation has also been applied in various NLP tasks. Earlier approaches were task-specific where the student model was distilled from a teacher model on some narrow downstream task. Tang et al. (2019) distilled BERT finetuned on paraphrasing and natural language inference tasks into a small BiLSTM model. Clark et al. (2019b) finetune BERT models on each of the GLUE tasks and then distill the multiple teacher models into a single multi-task student model. Yang et al. (2019) apply a similar approach of multi-teacher distillation for Question Answering. Tsai et al. (2019) employ knowledge distillation in a multilingual context by transferring knowledge from a larger multilingual BERT model to a smaller student model for sequence labeling tasks. Mohammadshahi et al. (2022) apply knowledge distillation to compress a machine translation model covering 100 languages.

Another line of work focuses on distilling task-agnostic models such as distilling large monolingual BERT models to pretrain smaller student BERT models for compression. Sanh et al. (2019) made a smaller version of BERT called DistilBERT by simplifying the original BERT. DistilBERT has the same basic architecture, but with half the number layers, and it starts with weights borrowed from the teacher model (BERT). They use knowledge distillation on the masked language modeling task and include the Cosine embedding loss in the training process. The student model is trained on the same dataset as the original BERT model but follows the pretraining procedure of Liu et al. (2019) by dropping the NSP objective, training on large batches, and using dynamic masking. They evaluate it on GLUE (Wang et al., 2018) and find that they retain 97% of the performance of BERT while being 40% smaller.

Followup work focuses on distilling not just the final teacher logits but also other components of the teacher. Sun et al. (2019) use an additional mean squared loss between

the hidden state output of the [CLS] token between either all layers or the last-k of the student and teacher BERT models. Jiao et al. (2020) extend this mean squared error loss to attention layers, hidden layers, embeddings, and outputs from the student and teacher models as seen in Figure 7. They choose the attention layer to transfer the linguistic knowledge captured in attention layers of teacher models (Clark et al., 2019a) to the student. The distillation is also performed at both the student pretraining phase as well as downstream task finetuning phase. Sun et al. (2020), unlike prior work that reduces the number of layers in the student model, instead reduces the width of the hidden layers to 128 and projects it to 512 for comparison with the teacher model.

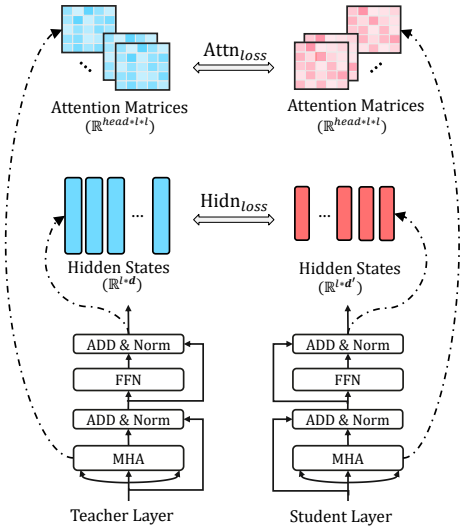

Figure 7: Knowledge Distillation from both attention and hidden layers. Figure reprinted from Jiao et al. (2020)

3.5 Huggingface Ecosystem

Huggingface provides a collection of open-source libraries for working with Transformer models and is widely used in the field. Since we make use of these in our work, we provide an overview of these libraries for the readers in this section.

Huggingface Transformers (Wolf et al., 2020) is their Python library to pre-train various transformer models from scratch as well as fine-tune existing pre-trained models easily

on downstream tasks. It supports a wide range of modalities from text, audio, and vision to multimodal models as well. The library is also framework-agnostic and supports both PyTorch and TensorFlow.

Tokenizers [4] provides access to subword tokenization algorithms such as BPE (Sennrich et al., 2016), WordPiece (Schuster and Nakajima, 2012; Wu et al., 2016) and Unigram. It also allows loading pre-trained tokenizers for various models as well as training a tokenizer from scratch on a new text corpus. Since the library is implemented in a low-level language like Rust with bindings in Python, the tokenization is very fast while still being easy to use.

Evaluate[5] is a library designed to assess NLP models that have been fine-tuned for different tasks. It gives you access to a range of standard machine learning metrics like accuracy, precision, recall, and f1-score, as well as specialized metrics tailored for specific tasks, such as BLEU score and Perplexity.

Datasets (Lhoest et al., 2021) provide access to popular machine learning datasets as well as support loading custom datasets from a wide range of file formats. The library provides capabilities such as applying batch operations on a dataset which makes use of multiple processes and is thus faster.

[4] `https://github.com/huggingface/tokenizers`

[5] `https://github.com/huggingface/evaluate`

4 Literature Review

In this chapter, we review the existing literature relevant to our research goals. Section 4.1 describes the curse of multilinguality associated with multilingual models and we explain in Section 4.2 how that has led to language-specific monolingual models being trained for different languages. Section 4.3 describes how knowledge distillation has been applied to train specific multilingual models for high-resource languages to get the benefits of both monolingual and multilingual models.

4.1 The Curse of Multilinguality

The curse of multilinguality refers to this phenomenon where fixed-capacity multilingual models trained in a large number of languages may not perform as well in individual languages as monolingual models trained in those languages. Conneau et al. (2020) studied this phenomenon when they proposed their multilingual model XLM-R and showed several sets of challenges associated with training multilingual models.

First of all, these models must accommodate a wide range of languages, each with its unique linguistic characteristics, vocabulary, and syntactic structures. This makes it challenging to optimize the model for the specific nuances of any single language.

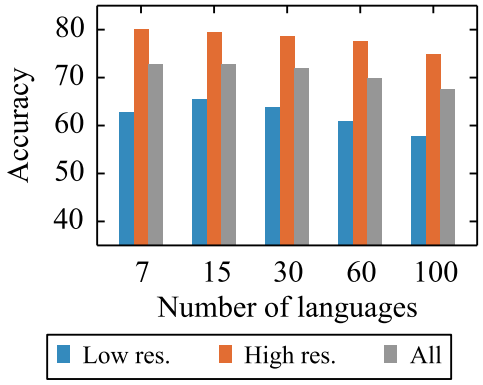

Figure 8: Impact of the number of languages on XNLI accuracy. Figure reprinted from Conneau et al. (2020)

As shown in Figure 8, the authors trained various multilingual models. They kept the

model size the same but increased the number of languages. They found that when they went from 7 to 15 languages, the model did better for low-resource languages, which is good. But when they added more than 15 languages, the performance dropped for both low and high-resource languages.

Balancing support for many languages with a manageable model size is tricky. This balance affects the model's ability to understand specific details in each language. In Figure 9, they noticed that when they kept the model size the same and added more languages (going from 7 to 30), performance suffered. But if they made the model a bit bigger (using a larger hidden size), the performance drop was not as bad. However, even with a larger model (hidden dimension of 1152), it didn't fully solve the challenges when dealing with 100 languages.

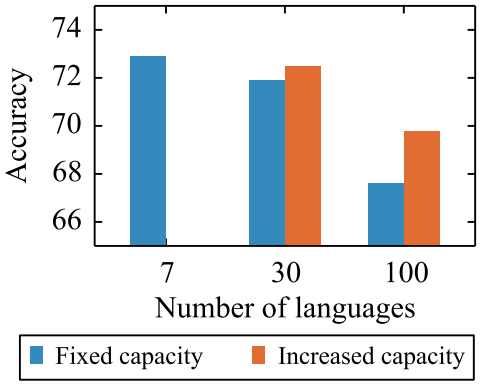

Figure 9: Tradeoff between model capacity and per-language performance. Figure reprinted from Conneau et al. (2020)

Another big problem is that there's not enough data available for low-resource languages compared to high-resource ones on the internet. This can cause a problem where the model does well with languages that have a lot of data but struggles with languages that don't. In Figure 10, they showed that if you increase the value of α (which means you use more data from high-resource languages), it helps the high-resource languages but hurts the low-resource ones. So, there's a trade-off, and you need to pick a balance. Conneau et al. (2020) discovered that setting α to 0.3 works well, giving some attention to low-resource languages during training without hurting high-resource languages.

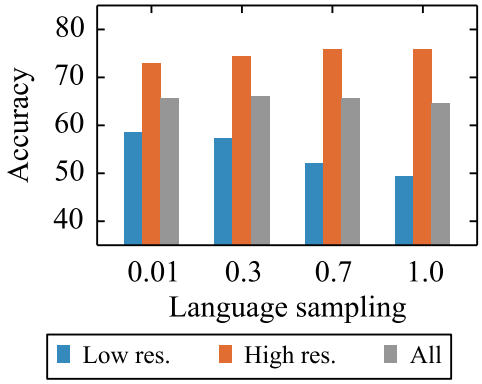

Figure 10: Impact of sampling factor on XNLI accuracy. Figure reprinted from Conneau et al. (2020)

Pfeiffer et al. (2022) demonstrate the curse of multilinguality in terms of the average language model perplexity getting worse when the number of languages is increased for a regular multilingual model, indicated by `shared` in Figure 11.

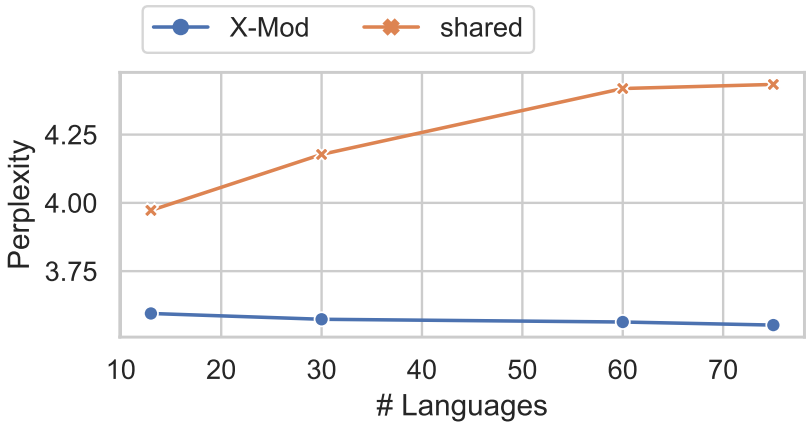

Figure 11: Impact of an increasing number of languages on average perplexity. Figure reprinted from Pfeiffer et al. (2022)

4.2 Rise of Language-specific BERT Models

The curse of multilinguality has given rise to the development of language-specific BERT models to improve the performance on tasks for that language even when they are supported in publicly available multilingual models. A common theme is to train these language-specific models on large amounts of web-scraped text corpora as well as use domain knowledge for specific components such as tokenization.

Antoun et al. (2020) trained a BERT-based model from scratch for Arabic called AraBERT and achieved state-of-the-art results beating mBERT on Arabic benchmarks. To achieve this, they built a large corpus of 24GB by merging texts scraped from Arabic news sites with previously available datasets (El-khair, 2016; Zeroual et al., 2019) and pre-trained BERT on that. They also leveraged domain knowledge of the language structure and segmented Arabic words with Farasa (Abdelali et al., 2016) before learning the tokenization from the corpus.

Similarly, Schweter (2020) trained an Italian BERT model by pre-training on a larger 13GB corpus formed by combining Italian Wikipedia with OPUS (Tiedemann, 2012) and achieved performance better than mBERT.

Nozza et al. (2020) survey monolingual BERT models from 18 languages. They found that these models mostly used three main sources for their training data. One source is Wikipedia, which has stuff in 301 languages. Another source is OPUS(Tiedemann, 2012), with parallel texts in 90 languages. The third source is the OSCAR(Ortiz Suárez et al., 2019) corpus, which has text in 166 languages. Often, the models are trained using a mix of data from these sources(de Vries et al., 2019).

This rise of these monolingual BERT models comes at a disadvantage in that they can't leverage the cross-lingual transfer benefits of multilingual training, especially from closely related high-resource languages. It also takes more storage and computing resources to host separate models for each language.

4.3 Knowledge Distillation as a Middle Ground

Khanuja et al. (2021) propose a framework called **MergeDistill** to leverage the idea of Knowledge Distillation to merge pre-trained monolingual BERT-based models for different languages into a single multilingual model. This provides a good compromise between monolingual models and massively multilingual models as you get the benefits of both approaches.

They set up this by leveraging pre-trained monolingual BERT models for different languages as multiple teachers and training a student BERT-based model to perform masked language modeling on a combined Wikipedia corpus for all those languages along with soft labels from the teachers.

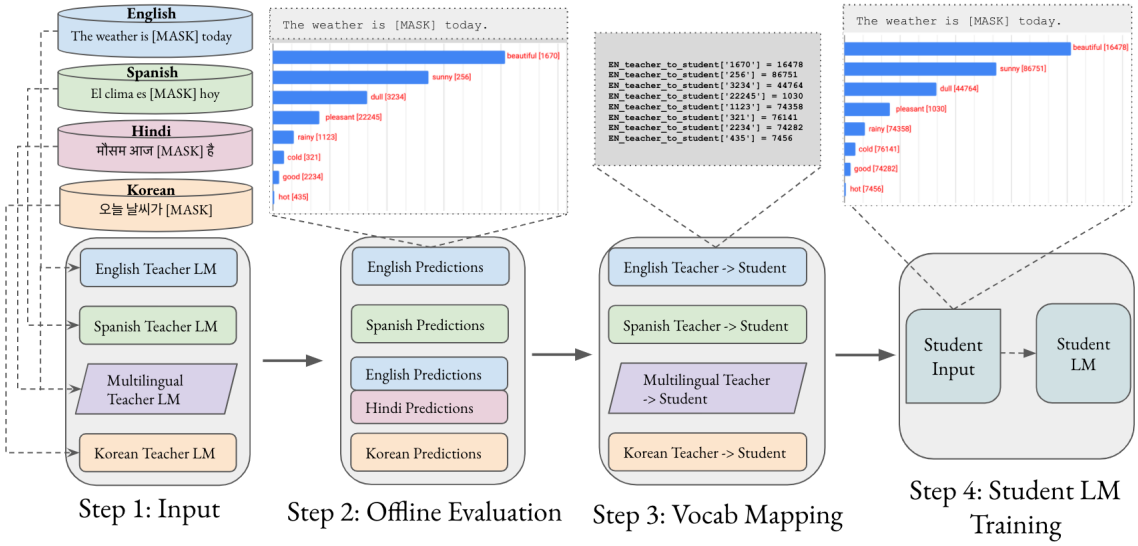

Figure 12: Overview of the MergeDistill Framework. Figure reprinted from Khanuja et al. (2021).

Figure 13 shows of how a piece of English text would get preprocessed for distillation. As seen, first the English text is tokenized into subwords using the tokenizer of the teacher model. Then 15% of the subwords are randomly masked. In the example, the word 'variety' is replaced with `[MASK]` token. The masked tokens are mapped to the vocabulary index in the teacher tokenizer. Then, the input is passed to the teacher model and the top-8 predictions from the teacher model are taken. In the figure, the logits before the softmax is applied are saved instead of saving the actual probabilities. Also, the

vocabulary indices for the top-8 masked token predictions are saved offline. Since the teacher and the student models would have different vocabularies, a mapping is done from the teacher to the student vocabulary for both the input indices and the predicted mask token indices. Once this step is completed, we get the data ready for training the student model for that particular teacher.

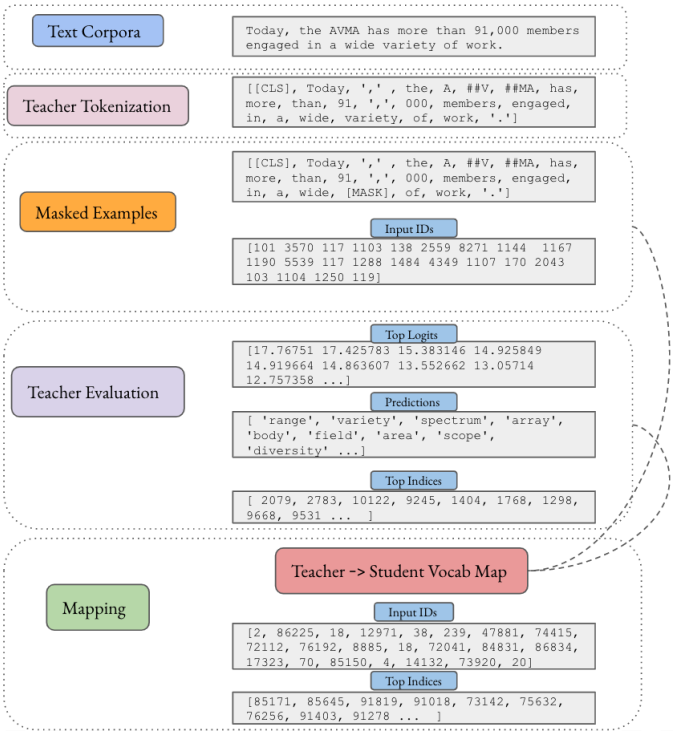

Figure 13: Generation of Teacher Predictions for Distillation in MergeDistill. Figure reprinted from Khanuja et al. (2021).

Figure 12 shows the entire flow for the MergeDistill process for four different teachers. We repeat the steps seen in Figure 13 for English, Spanish, Korean, and a Multilingual Teacher LM. All the predictions from these teacher models are mapped to the student vocabulary and we combine the offline datasets and shuffle them to create examples from multiple languages in the same batch. Once this offline dataset is ready, the student model is trained using the MLM objective on this dataset.

They found that when merging languages from a similar family, the per-language performance of the merged student model was on par or better than the monolingual model on downstream tasks. They test this on four related languages - English, German, Ital-

ian, and Spanish. The resulting student model outperformed the individual monolingual BERT models on NER and QA while being on par in dependency parsing. When training on five dissimilar languages – Arabic, Chinese, English, Finnish, and Turkish, the resulting multilingual model performs worse than the individual monolingual models on all tasks as expected.

They also show that the downstream performance improves when using teacher model predictions for distillation in a limited data scenario.

4.4 Takeaways

As seen in Section 4.3, MergeDistill is a promising direction for data-efficient pretraining. The languages that Khanuja et al. (2021) chose in their experiments are high-resource and have comparatively larger corpus in Wikipedia. However, there are languages such as Maltese whose entire Wikipedia only consists of around 5 thousand articles. It can be a promising direction to evaluate the MergeDistill framework in a low-resource language setting and we pursue that in this thesis work for Maltese by leveraging Arabic, English, and Italian as the related languages.

Maltese is particularly interesting because it is a low-resource language (Rosner and Borg, 2022) and it's not even included in popular multilingual models like mBERT, XLM, and XLM-R. Those models are trained for only the most common languages.

A recent work by Micallef et al. (2022) pre-trained the first monolingual BERT model for Maltese from scratch called BERTu. Given the small size of Maltese Wikipedia, they instead collected data and expanded an existing corpus of unlabeled text called Korpus Malti v3.0 [6] to create a new text corpus called Korpus Malti v4.0. In addition to a monolingual model, Micallef et al. (2022) also trained a multilingual model called mBERTu by further pretraining the mBERT model on Maltese following the procedure done in Chau et al. (2020) and Muller et al. (2021).

They evaluated both models on downstream tasks for Dependency Parsing, Part-of-Speech Tagging, Named-Entity Recognition, and Sentiment Analysis. They find that the monolingual model BERTu performs better than multilingual mBERTu on all tasks

[6] `https://mlrs.research.um.edu.mt/`

except Part-of-Speech Tagging.

Given the availability of a strong monolingual teacher model BERTu for Maltese that has been trained on a larger corpus, MergeDistill is applicable. We can combine BERTu with the pre-trained models for related languages – Arabic, Italian, and English using knowledge distillation and compare its performance on the downstream tasks. We do this using only the 5.1K Wikipedia articles as the pretraining corpora for Maltese to see how effective MergeDistill style pretraining is in a data-scarce setting compared to pretraining from scratch. We can also evaluate the relative performance compared to a monolingual and multilingual Maltese model that has been trained on a far larger corpus.

5 Experimental Framework

In our work, we adapt the methodology introduced in MergeDistill (Khanuja et al., 2021) to distill BERT-based models from languages related to Maltese together with a monolingual Maltese model BERTu to create a specific multilingual Maltese BERT model. In this chapter, we explain the various steps involved in this along with implementation details and choices made. We reimplemented the whole framework from scratch based on the details provided in Khanuja et al. (2021) since their codebase isn't publicly available.

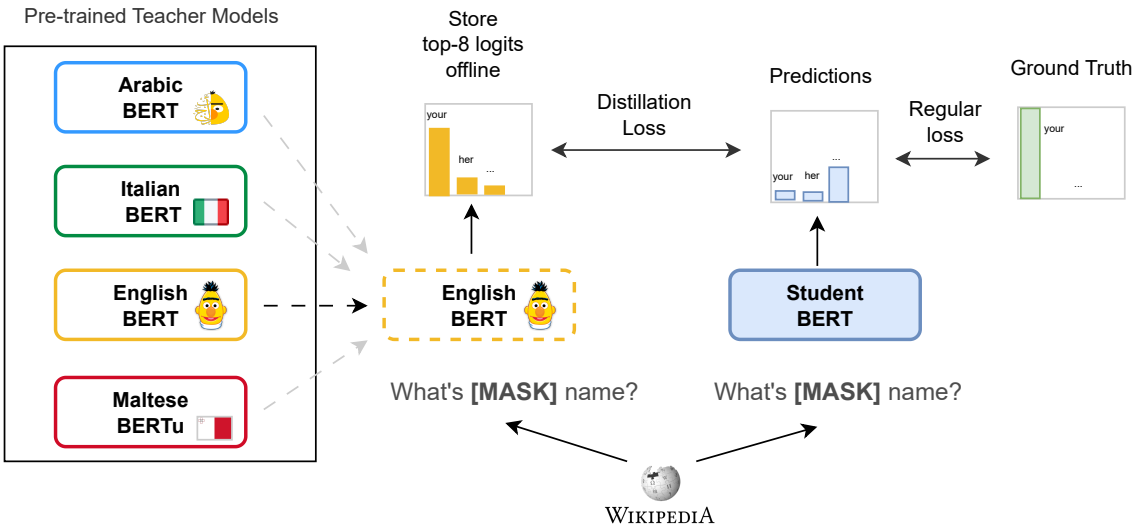

Figure 14: Overview of the Experimental Framework

5.1 Pre-training Data Collection

In our work, we use languages related to Maltese, namely English, Italian, and Arabic, and thus in addition to Maltese pre-training data, we will require text corpora for these related languages for distillation. Each corpus can be different from the dataset used to originally pretrain the monolingual BERT-based models for those languages. We chose Wikipedia pages for each language as our primary data source because they covered diverse topics and domains. This controls for different domains between the languages and allows us to make a more fair comparison.

Wikipedia provides dumps of all their articles for each language periodically [7]. These

[7] https://dumps.wikimedia.org/

dumps are raw articles with all Wikipedia-specific tags and references and require additional preprocessing. We used the export file with articles written up to May 1, 2023.

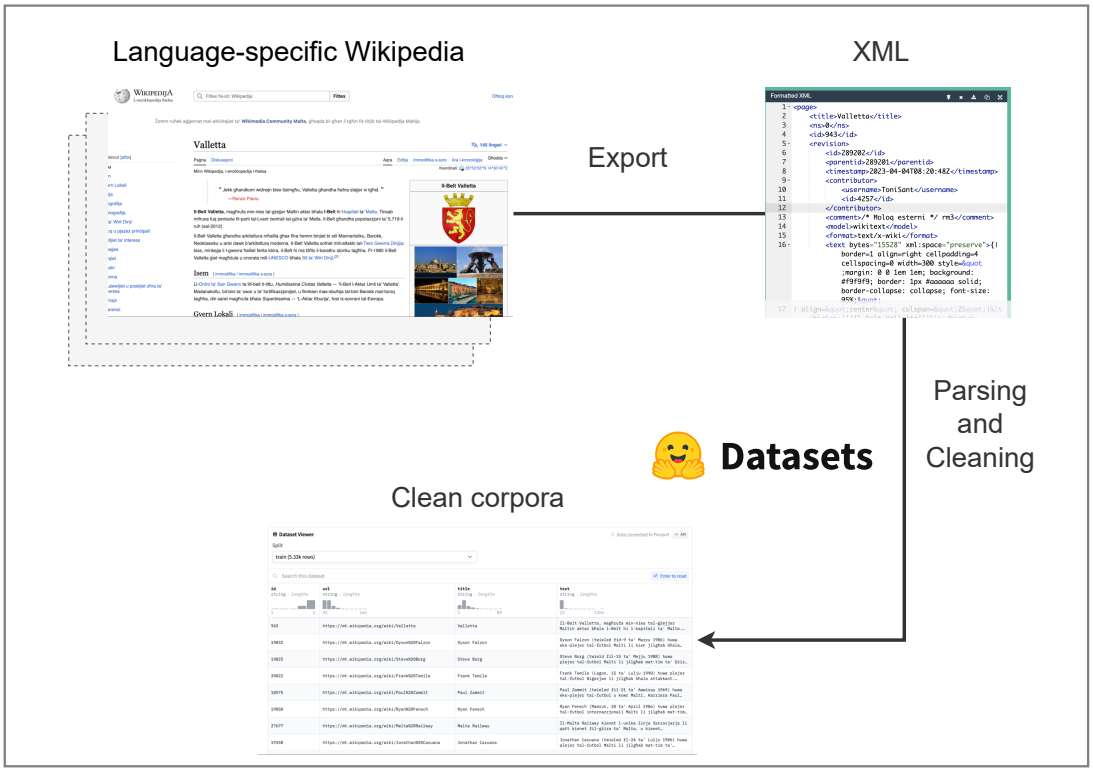

Figure 15: Pipeline for collection of texts corpora from Wikipedia

As seen in Figure 15, we preprocess the XML file exported from Wikipedia by using the script [8] provided out of the box in Huggingface datasets API [9] for Wikipedia. A preprocessed version of the English Wikipedia corpus is already available in the library. For Arabic, Italian, and Maltese, we ran the pre-processing script on a compute cluster on our end. This was executed on a compute cluster with 32 CPU cores and a total memory of 256GB and the generated datasets were saved.

Under the hood, the datasets (Lhoest et al., 2021) library uses an open-source tool mwparserfromhell[10] to parse the Wikipedia files and extract clean text corpus that is ready for model training. The library uses Apache Beam[11] to parallelize the pre-processing when

[8] https://huggingface.co/datasets/wikipedia/blob/main/wikipedia.py

[9] https://huggingface.co/datasets/wikipedia

[10] https://mwparserfromhell.readthedocs.io/en/latest/index.html

[11] https://beam.apache.org

run on a cluster.

Table 1 shows the article statistics for the four languages after pre-processing. As seen, there is a huge imbalance between high-resource languages like English compared to Maltese.

Language	**Article Count**	**Percentage**
English	6.62 million	65.94
Italian	1.8 million	17.91
Arabic	1.6 million	16.09
Maltese	5172	0.05

Table 1: Distribution of Wikipedia articles by language

5.2 Offline Generation of Predictions from Teacher Models

Once our text corpus was ready, we first prepared the corpus to a format suitable for pre-training a masked language model.

For the corpus of each language, we use the respective teacher BERT model and tokenizer. We take pre-trained AraBERT (Antoun et al., 2020), ItalianBERT (Schweter, 2020), BERT (Devlin et al., 2018), and BERTu (Micallef et al., 2022) models for Arabic, Italian, English, and Maltese respectively as the teacher models. These pre-trained models are available on the Huggingface Hub and can be accessed through the transformers (Wolf et al., 2020) library.

For each language's text, we break it into chunks of 512 tokens, following BERT(Devlin et al., 2018). Then, in each chunk, we randomly hide 15% of the tokens. Among these hidden tokens, 80% turn into a [MASK] token, 10% become a random token from the vocabulary, and 10% stay as they are. We also store what these hidden tokens originally were as labels.

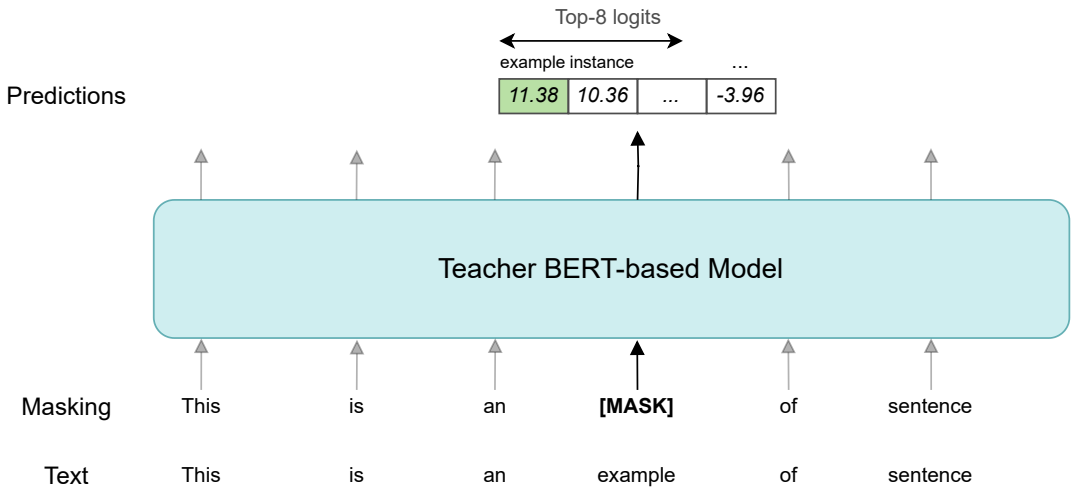

Figure 16: Generation of masked token predictions from the Teacher Model

As seen in Figure 16, we then run inference using the respective teacher model. We pass the segments to the teacher model in batches and get the masked token predictions as logits over the entire vocabulary. Since vocabulary sizes range from 28k to 64k for the different models, storing the predictions on the entire vocabulary for each masked token would take a lot of storage and is infeasible. Thus, we only store the top 8 logits for each masked token following Khanuja et al. (2021). From this procedure, we have the teacher's predictions for each masked token in the entire corpus.

We also perform additional optimizations to reduce the storage space needed for these preprocessed logit datasets. Each input sequence can be 128 or 512 tokens in length but we only care about the masked token positions. Thus, we optimize storage by storing the logits only for the masked token positions instead of logits for the entire 512 tokens in the sequence. We recover the structure back to 128 tokens or 512 tokens during training time. The logits are stored offline as a huggingface dataset, which converts the data to the optimized parquet format and also shards the large dataset into multiple files. The storage required for the different languages is presented in table 2.

Language	Teacher Model	Number of Sequences	Storage
Maltese	BERTu	12,700	73.2 MB
Arabic	bert-base-arabert	1.06 million	6.84 GB
Italian	bert-base-italian-cased	2.06 million	13.7 GB
English	bert-base-cased	8.62 million	56.9 GB

Table 2: Offline Logits per Language

5.3 Merging Teacher Tokenizers

All four pre-trained models for Italian, English, Maltese, and Arabic have pre-trained WordPiece (Schuster and Nakajima, 2012; Wu et al., 2016) tokenizers available. To create a student tokenizer, we take the union of the vocabulary of the teacher WordPiece tokenizers. Since WordPiece tokenizers work based only on the vocabulary and don't store the merge rules like BPE (Sennrich et al., 2016), they are simpler and trivial to merge.

Below, we show an example of tokenization on a text that is code-mixed between English and Maltese. A tokenizer from a monolingual Maltese model like BERTu doesn't have the word 'Hello' in its vocabulary and thus would break the word 'Hello' into subwords 'Hel' and `'##lo'`. Similarly, a tokenizer for English BERT has 'Hello' in its vocabulary but breaks the Maltese word 'Kif' into subwords 'Ki' and `'##f'` and the word 'int' into subwords 'in' and `'##t'`. But a student tokenizer created from a combination of both BERTu and bert-base-cased was able to tokenize both languages as seen in the example.

Original Text: *Hello! Kif int?*
Maltese (BERTu):: *Hel, ##lo, !, Kif, int, ?*
English (BERT):: *Hello, !, Ki, ##f, in, ##t, ?*
Maltese + English (mergedistill-mt-en): *Hello, !, Kif, int, ?*

Figure 17 shows the percentage overlap between the vocabulary of the tokenizers for Arabic(ar), English(en), Italian(it), and Maltese(mt). There is an overlap of around 10% between all combinations of English, Italian, and Maltese. Arabic, on the other hand,

has less than 2.5% overlap with the rest of the languages, which is not surprising, given that it is written in a different script compared to the Latin script used by the rest of the three languages.

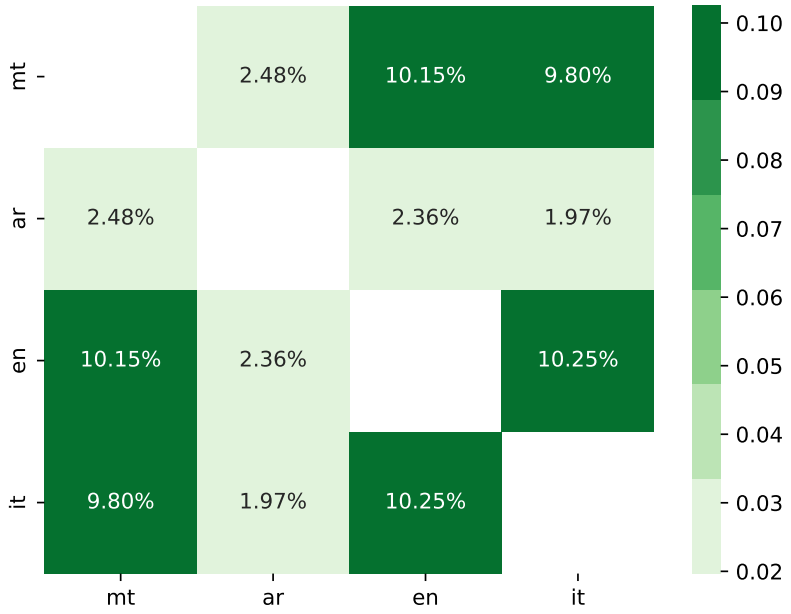

Figure 17: Percentage of vocabulary overlap between tokenizers. The diagonal is not shown in the heatmap since it would amount to 100%

Table 3 shows the vocabulary size for the tokenizer of the teacher model for different languages and the vocabulary size of the resulting student tokenizer from the combinations. Since both English and Italian are written in the Latin script, they have a larger vocabulary overlap with Maltese compared to Arabic.

The 2.48% overlap between Maltese and Arabic which amount to 2810 tokens were things like punctuations (e.g. *!, ”*), numbers, and English subwords (*##struction, China* etc.). Since AraBERT (Antoun et al., 2020) used a pretraining corpus extracted from news portals, they have code-mixing of English and Arabic. News articles, especially in categories such as international news make use of non-native named entities as well as technical terms in more scientific domains. Antoun et al. (2020) points out in the original paper that they didn't implement any word-level language filtering and instead preserved words with Latin characters because of this reason.

Language	Vocab Size	Overlap %
English (bert-base-cased)	28,996	
Italian (bert-base-italian-cased)	31,102	
Maltese (BERTu)	52,000	
Arabic (bert-base-arabert)	64,000	
Maltese + Arabic	113190	2.48%
Maltese + Italian	75,688	9.80%
Maltese + English	73,533	10.15%

Table 3: Vocabulary Sizes of different language combinations

When we mix tokenizers using the vocabulary union method, there's a limitation: it only works when the models use the same subword tokenizer. For instance, we can't merge a model like RoBERTa(Liu et al., 2019), which uses a byte-pair encoding (BPE)(Sennrich et al., 2016) tokenizer, with a BERT-style model that relies on the WordPiece tokenizer. To use our method, all teacher models must use the WordPiece tokenizer.

5.4 Combining Logits from Multiple Languages

Once we generate the logits from the teacher model of each language, we create different combinations of language with Maltese by simply concatenating the generated data. We also shuffle the data so that sequences from different languages are present in the same batch during pretraining. For example, we can create data to train a mergedistill-mt-it model by merging logits data for Maltese and Italian generated from the procedure described in step 5.2.

As seen in Table 2, Wikipedia's coverage across the four languages varies significantly. Training on the same distribution could result in low-resource languages like Maltese being inadequately represented. To solve this, we follow exponential smoothing (Devlin et al., 2019) to undersample high-resource languages and oversample low-resource languages based on the number of sequences in each language.

Given that we have N different languages, the original probability distribution of the

number of sequences in each language $\{p_i\}_{i=1...N}$ is given by:

$$p_i = \frac{n_i}{\sum_{k=1}^{N} n_k} \text{ (Lample and Conneau, 2019)}$$

We generate a new distribution $\{q_i\}_{i=1...N}$ by smoothing the original probabilities with α and re-normalizing it. This new distribution is then used to sample the sequences from the various languages during training.

$$q_i = \frac{p_i^{\alpha}}{\sum_{j=1}^{N} p_j^{\alpha}} \text{ (Lample and Conneau, 2019)}$$

We use an α value of 0.7 following Khanuja et al. (2021) for exponential smoothing. A higher α value favors high-resource languages and vice versa. Figure 18 shows the original distribution of the number of samples in the four languages. After applying exponential smoothing, we see that Maltese, Italian, and Arabic are sampled more often compared while English is downsampled.

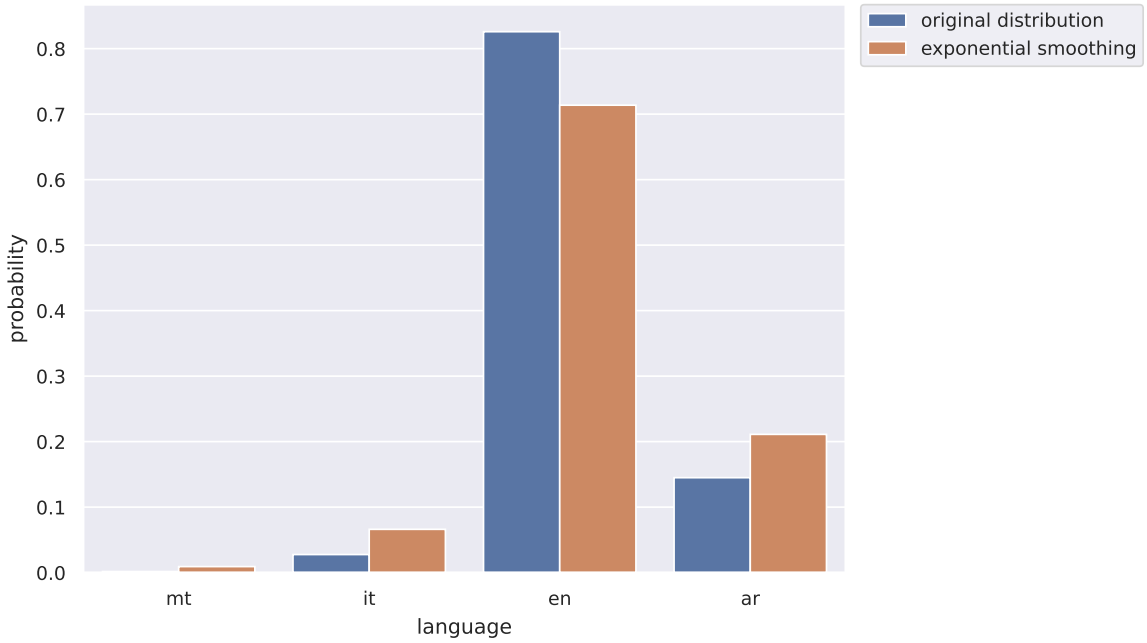

Figure 18: Impact of Exponential Smoothing on Language Sampling Distribution

5.5 Language Model Pre-training with Distillation

With all the data ready, we can train the student BERT model. Our student BERT model uses the same base architecture as the bert-base-cased. This BERT architecture has 12 encoder layers with 12 attention heads, an embedding dimension of 768, and supports a maximum sequence length of 512 tokens. The vocabulary size depends on which monolingual language models we are merging from Table 3.

We leverage the Huggingface transformers library for the training. The library provides built-in utilities to train masked language models like BERT. However, it doesn't support distillation. Therefore, we override their `Trainer` class to use a custom loss function for knowledge distillation. In addition to a regular cross-entropy loss to compare model predictions for masked tokens to the ground-truth token index, another cross-entropy loss is added to compare the student model predictions to the teacher model predictions for the same masked token.

Mathematically, given a student model with a vocabulary of size v that gets an input with n masked tokens, we try to predict the $x_m = \{x_1, x_2, x_3 ... x_n\}$ masked tokens when given the x_{-m} non-masked tokens. To train this, we use cross-entropy loss with one-hot-encoded labels, as seen in the equation below:

$$L_{MLM}(x_m|x_{-m}) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{|v|} P(x_{m_i}, k) \text{ (Khanuja et al., 2021)}$$

$$P(x_{m_i}, k) = \mathbf{1}(x_{m_i} = k) \log p(x_{m_i} = k|x_{-m}; \theta) \text{ (Khanuja et al., 2021)}$$

In addition to the MLM loss used in regular BERT training, we also compare masked token predictions to the predictions we generated offline from the teacher models. The output probability distribution for a masked token x_{m_i} from the teacher model is denoted by $Q(x_{m_i}|x_{-m}; \theta_t)$. Thus, the knowledge distillation loss L_{KD} is the cross-entropy loss between the teacher and the student distributions.

$$L_{KD}(x_m|x_{-m}) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{|v|} \hat{P}(x_{m_i}, k) \text{ (Khanuja et al., 2021)}$$

$$\hat{\mathrm{P}}(\mathrm{x_{m_i}}, \mathrm{k}) = \mathrm{Q}(\mathrm{x_{m_i}} = \mathrm{k}|\mathrm{x_{-m}}; \theta_\mathrm{t})\ \mathrm{logp}(\mathrm{x_{m_i}} = \mathrm{k}|\mathrm{x_{-m}}; \theta)\ \text{(Khanuja et al., 2021)}$$

The final loss is a weighted sum of these two losses and the weight for the knowledge distillation loss is denoted by λ and has a value between 0 and 1.

$$\mathrm{L_{ALL}} = \lambda \mathrm{L_{KD}} + (1 - \lambda)\mathrm{L_{MLM}}\ \text{(Khanuja et al., 2021)}$$

The value of λ is decayed during training using teacher annealing (Clark et al., 2019b) as seen in Figure 19. In our implementation, λ is set to the percentage of training remaining expressed as a floating point number between 0 and 1 and calculated from the training steps done and total training steps. At the beginning of the training, $\lambda = 1$ and thus the student model only learns from the teacher model predictions using distillation. As the training progresses, the loss will be a combination of both distillation and regular MLM loss. Toward the end of the training, the model learns only from supervised ground-truth labels instead of teacher logits. Hence, the model training gradually shifts from knowledge distillation to supervised finetuning as the training progresses.

Figure 19: Decay of the weight λ given to the teacher loss as the training progresses

Khanuja et al. (2021) pre-trained their student model for 250K steps with a batch size of 4096. However, given computational constraints, we instead use a scaled-down batch

size of 128 for 250K steps using 8 A100-40GB Nvidia GPUs on a SLURM (Yoo et al., 2003) cluster. This is only 3.125% of what the original authors trained for.

We use a learning rate of 1e-4, weight decay of 0.01, and a warm-up ratio of 1% during the pretraining. We make use of AdamW (Loshchilov and Hutter, 2019) as the optimizer and also apply mixed precision to speed up the training process.

6 Evaluation Criteria

To evaluate our multi-teacher distillation based pretraining approach, we evaluate the student model on various downstream tasks used in prior work (Micallef et al., 2022). The downstream tasks were chosen so that we evaluate the capabilities on both semantic tasks like sentiment analysis and news categorization as well as low-level tagging tasks like part of speech tagging and named entity recognition.

Below, we describe the downstream tasks and how the pre-trained BERT-based architectures are modified for these various downstream tasks:

Sentiment Analysis: We use the Maltese sentiment analysis dataset (Martínez-García et al., 2021). This dataset was created by consolidating two existing sentiment dataset –Dingli and Sant (2016) who annotated sentiment for texts from Maltese microblogs and Cortis and Davis (2019) who annotated sentiments for social media texts centered on a the topic of government budget. The task is to classify a given piece of Maltese text into positive and negative and thus this is a binary classification problem.

For the sentiment analysis task, we take the output of the [CLS] token in the last transformer layer, add dropout, and then use a linear layer to generate logits for two classes. A regular cross-entropy loss is used as the loss function.

We use accuracy as the evaluation metric which is defined by:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

News Tagging: This semantic task involves tagging the contents of a news article with multiple relevant tags among 17 different tags such as 'International', 'Culture' etc. We evaluate on a multi-label classification problem of News Tagging. This is a newly created dataset and has not been evaluated in prior work. Details on how this dataset was created are included in the appendix A.1.

Since this is a multi-label classification problem, we take the output of the [CLS] token

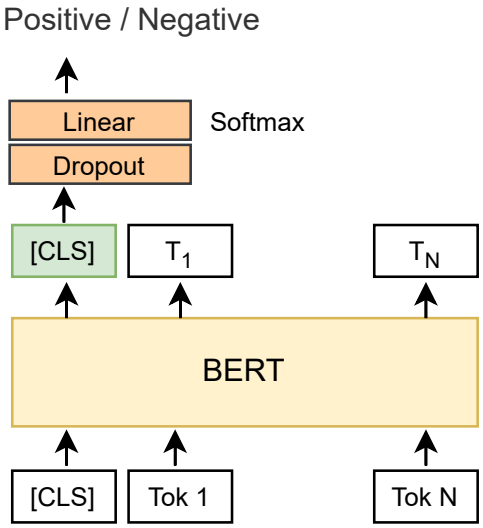

Figure 20: Architecture for the Sentiment Classification Task

in the last transformer layer, add dropout, and then use a linear layer to generate logits for the output tags. We use the sigmoid activation function to the final logits to generate separate predictions for all the tags and use a binary cross entropy loss for comparison against ground-truth labels.

We use Macro F1 as the evaluation metric.

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1 = \frac{2 \cdot \text{Precision} \cdot Recall}{\text{Precision} + Recall}$$

Part of Speech(POS) Tagging: We use the MLRS POS data (Gatt and Čéplö, 2013) for part-of-speech tagging. The original version of the dataset had human-annotated sentences categorized into eight domains. Micallef et al. (2022) combine and shuffle these data from different domains and generate a new split of 80% training, 10% validation, and 10% testing sets. In our work, we follow the same splits as Micallef et al. (2022).

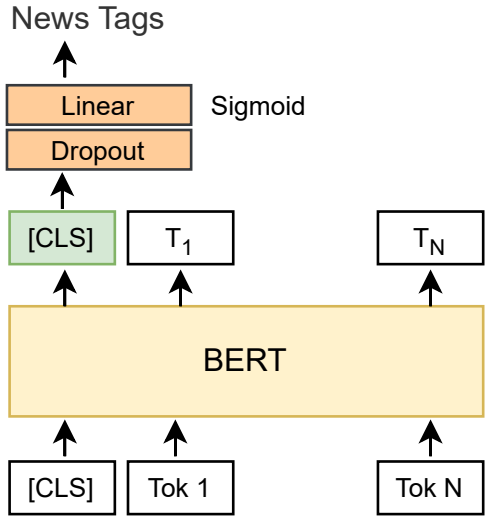

Figure 21: Architecture for the News Tagging Classification

For this token classification task, we use dropout followed by a linear layer applied across the 768-dimensional output for each token in the last transformer layer. Since the tokens generated by the BERT tokenizer can include subwords, we repeat the same POS label for words that have been tokenized to multiple subwords.

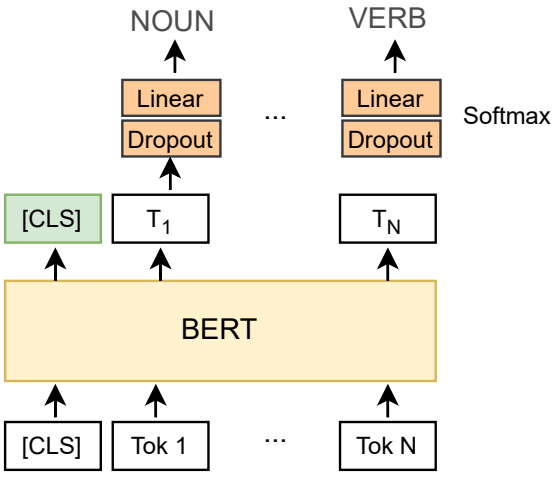

Figure 22: Architecture for the POS Tagging Task

We use seqeval (Nakayama, 2018) to calculate the accuracy.

Named Entity Recognition (NER): We use the WikiAnn data (Pan et al., 2017) for named entity recognition. This dataset was created from Wikipedia using an automated knowledge base mining approach.

As this is also a token classification task, we finetune it in a similar fashion to POS tagging by using dropout followed by a linear layer applied across the 768-dimensional output for each token in the last transformer layer. For words that have been tokenized to multiple subwords, we use the same named entity label for the subwords as the entire word. However, we assign a B-X token for the first subword to denote the beginning of an entity span and then use the I-X to indicate that the following subwords are inside, where X denotes the entity category like PER (person), LOC (Location), etc.

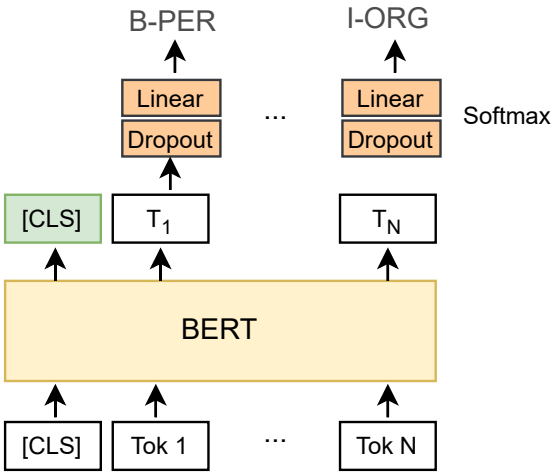

Figure 23: Architecture for the NER Task

We use span-based F1 score as our evaluation metric and use seqeval (Nakayama, 2018) to calculate it.

Table 4 summarizes the size of the datasets for our chosen downstream tasks across various splits as well as the evaluation metrics and the source domain from where the dataset was created.

Our student model is finetuned on each of these tasks and the metric for each task is

Dataset	|Train|	|Validation|	|Test|	Metric	Domain
News Tagging	10.8k	2.3k	2.3k	Macro F1	news
POS Tagging	4.94k	616	616	Accuracy	mixed
Sentiment Analysis	595	85	171	Accuracy	blogs/social media
NER	100	100	100	Span-based F1	Wikipedia

Table 4: Dataset size and Evaluation metrics for the various downstream tasks

compared to BERTu/mBERTu (Micallef et al., 2022) and baselines. As baselines, we train the monolingual teacher models ItalianBERT (bert-base-italian-cased), AraBERT (bert-base-arabert), and BERT (bert-base-cased) directly on the Maltese downstream tasks. This was done to get a baseline score for models that had not seen Maltese during their pretraining. Furthermore, we also use a BERT model with the same configuration as our student model but trained from scratch on the Maltese Wikipedia dataset instead of using distillation as an additional baseline.

Our finetuning procedure differs from Micallef et al. (2022) in that we use the Huggingface transformers (Wolf et al., 2020) library to finetune the BERT models on our downstream tasks while the authors used AllenNLP (Gardner et al., 2018). We made this choice since the development on the AllenNLP (Gardner et al., 2018) library has been discontinued and the library has been set to archive mode since Dec 16, 2022 [12]. As such, we finetuned both BERTu and mBERTu models on our downstream tasks for a fair comparison. We chose to use the Transformers library as that allows us to share the same codebase for both pre-training and finetuning. This also makes it easily reproducible as the transformers library is more frequently updated. We use the built-in architectures provided by the transformers library for multiclass classification, multi-label classification, and token classification tasks.

We follow the same training procedure for finetuning on downstream tasks as Micallef et al. (2022). The hyperparameters used for the different tasks as shown in Table 5. We

[12] https://github.com/allenai/allennlp/commit/20df7cdd3eea7f895ceee9c57e2be1a843510748

finetune each model for at most 200 epochs, with an early stopping on the validation set. The patience for Early Stopping is set to 20 epochs. We use five different random seeds for each task from 1 to 5 and report the mean and standard deviation of the score in our results.

Name	**Batch Size**	**Dropout**	**Learning Rate**
News Tagging	32	0.5	4e-5
Sentiment Analysis	32	0.5	2e-5
NER	64	0.2	2e-5
POS Tagging	128	0.3	2e-5

Table 5: Hyperparameters used for finetuning per task

7 Experiments

7.1 Knowledge Distillation from Maltese to Maltese

In this experiment, we wanted to first validate whether knowledge distillation works simply by using a monolingual Maltese model as a teacher model and distilling it into a monolingual Maltese student model. This acts as a baseline to compare the performance when using more than one language as a teacher for distillation.

Setup

We use BERTu (Micallef et al., 2022) as our teacher model and generate the masked token predictions on a Maltese Wikipedia dataset and store it offline. We use the same tokenizer as BERTu for the student model.

We train two models on this dataset. The first model is trained only on the ground-truth labels without any distillation loss from the teacher. This is equivalent to setting $\lambda = 0$ to the loss we described in the methodology section.

$$\mathrm{L_{ALL}} = \lambda \mathrm{L_{KD}} + (1 - \lambda)\mathrm{L_{MLM}} \text{ (Khanuja et al., 2021)}$$

When $\lambda = 0$:

$$\mathrm{L_{ALL}} = 0 * \mathrm{L_{KD}} + (1 - 0) * \mathrm{L_{MLM}}$$

$$\mathrm{L_{ALL}} = \mathrm{L_{MLM}}$$

As seen, the loss becomes equivalent to using a supervised masked language modeling loss objective without any distillation.

The other model is trained using both the losses and denoted by mergedistill-mt. We pre-train both the models for 30k steps instead of 250K steps to prevent overfitting.

Results and Discussion

When both models are trained for the same duration, the student model, which uses distillation, is better at making use of data and reaches a lower Masked Language Modeling (MLM) loss sooner in the training process compared to a model starting from scratch.

We can observe this around the 5,000 step mark on the graph. This supports our idea that knowledge distillation makes pre-training more efficient with data.

Furthermore, we also see that the regular model trained without distillation starts overfitting on the small-sized Wikipedia dataset and the evaluation loss starts increasing when trained for longer steps. While for a distilled model, we see the loss curve is more stable. This shows that knowledge distillation acts as a form of regularization during pretraining.

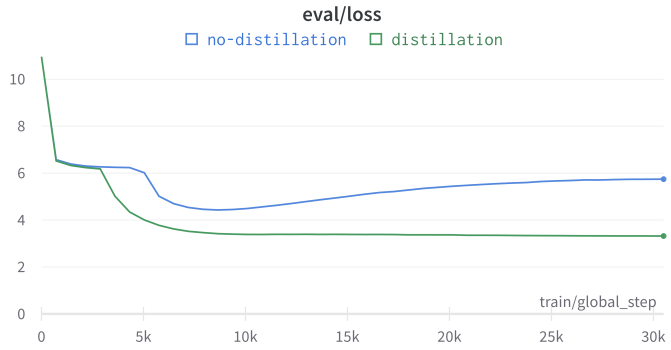

Figure 24: Evaluation Loss Curve for models trained with knowledge distillation vs without

Next, we also evaluate it on the downstream tasks and find that the student model trained with distillation performs better than a model trained from scratch across all the downstream tasks as seen in Table 6. The gap is more prominent in the named entity recognition task where the distilled model achieves an average Span F1 of 45.36 while a regular model achieves only a score of 33.52.

Model	Accuracy
no-distillation-mt	69.59 ± 1.01
mergedistill-mt	**70.76** ± 1.16

(a) Sentiment Analysis

Model	Macro F1
no-distillation-mt	56.22 ± 1.80
mergedistill-mt	**58.59** ± 2.24

(b) News Tagging

Model	Accuracy
no-distillation-mt	94.42 ± 0.16
mergedistill-mt	**95.76** ± 0.06

(c) POS Tagging

Model	Span F1
no-distillation-mt	33.52 ± 3.14
mergedistill-mt	**45.36** ± 4.84

(d) Named Entity Recognition (NER)

Table 6: Comparison of a student Maltese model trained from scratch vs knowledge distillation, grouped by the downstream task.

7.2 Distillation with Multiple Teachers

In this experiment, we merge a Maltese monolingual BERT model with each of the related languages, namely Italian, Arabic, and English, and check the performance impact compared to simply distilling from Maltese only. This should help us understand the impact of distilling from related languages.

Setup

For Maltese, we use BERTu as the monolingual teacher model and use ItalianBERT (bert-base-italian-cased), AraBERT (bert-base-arabert), and BERT (bert-base-cased) for Italian, Arabic, and English respectively. We generate the offline logits from each of these models on the respective language-specific Wikipedia beforehand as described in the Section 5.2 of Methodology.

We initialize a student model with the same architecture as BERTu (Micallef et al., 2022) but with random weights and the tokenizer is generated using the union of the vocabulary of the selected languages. The student model is trained using both ground-truth loss and distillation loss and then evaluated on downstream tasks. Each student model is trained for 250K steps.

As baselines, we also finetune the teacher model for English, Italian, and Arabic directly on the downstream tasks. These models have not been trained on Maltese and can act as a proxy to see how much performance lift we get when we merge them with a Maltese model.

Results and Discussion

The results of the downstream tasks are reported in Table 6. As seen, compared to a baseline student model (mergedistill-mt) that has only been distilled from Maltese as described in Section 7.1, all the student models distilled from a combination of two languages perform better than it across all the tasks. The only exception to this trend is the Maltese + English (mergedistill-mt-en) student model on the Sentiment Analysis task where its accuracy of 69.29 is lower than the baseline accuracy of 70.76 for a model only distilled from BERTu (mergedistill-mt).

We also see that the distilled student models perform better than their respective teacher models on all the downstream tasks. The only exception is in the News Tagging task where a Maltese + English (mergedistill-mt-en) model has a Macro F1 score of 61.21, which is lower than its teacher model bert-base-cased score of 61.51. We believe this is because of the way the dataset has been sourced. The three largest categories in the news dataset are 'International', 'Entertainment', and 'Sports' and as such, include a lot of named entities preserved in their original English script such as names of a sportsperson, celebrities, etc. The presence of such named entities could provide surface-level signals for classification into the respective category even for a model such as bert-base-cased that has not been trained on Maltese directly.

In terms of performance lift, we find that the NER task benefits the most from the distillation process. Compared to the baseline, we find a big jump in performance with all three distilled models: mergedistill-mt-it, mergedistill-mt-ar, and mergedistill-mt-en. We believe this is because the WikiANN dataset was created from Wikipedia and our distillation-based pre-training was also done on text corpora extracted from Wikipedia. This is consistent with Micallef et al. (2022) which also shows that there are performance gains when the pre-training domain matches the downstream task domain.

Combination	**Model**	**News Tagging** (Macro F1)	**Sentiment Analysis** (Accuracy)
Maltese	mergedistill-mt	58.59 ± 2.24	70.76 ± 1.16
Maltese + Italian	bert-base-italian-cased	61.20 ± 1.80	69.12 ± 2.81
	mergedistill-mt-it	62.64 ± 0.91	72.07 ± 1.87
Maltese + Arabic	bert-base-arabert	58.22 ± 1.90	68.42 ± 1.01
	mergedistill-mt-ar	**62.80** ± 1.99	**73.87** ± 1.21
Maltese + English	bert-base-cased	61.51 ± 0.87	66.43 ± 3.52
	mergedistill-mt-en	61.21 ± 1.33	69.29 ± 1.81

(a) Performance on semantic tasks

Combination	**Model**	**NER** (Span F1)	**POS Tagging** (Accuracy)
Maltese	mergedistill-mt	45.36 ± 4.84	95.76 ± 0.06
Maltese + Italian	bert-base-italian-cased	54.21 ± 2.51	91.98 ± 0.10
	mergedistill-mt-it	68.33 ± 3.16	**96.55** ± 0.08
Maltese + Arabic	bert-base-arabert	38.35 ± 2.13	91.35 ± 0.14
	mergedistill-mt-ar	66.30 ± 3.29	96.16 ± 0.11
Maltese + English	bert-base-cased	53.45 ± 5.23	95.96 ± 0.14
	mergedistill-mt-en	**72.64** ± 3.94	96.41 ± 0.11

(b) Performance on syntactic/token-level tasks

Table 7: Experimental results, grouped by the downstream task.

7.3 Comparison to Monolingual and Massively Multilingual Models

In this experiment, we compare the multi-teacher distilled models to a monolingual Maltese model BERTu, and a massively multilingual Maltese model called mBERTu.

Setup

We use the pre-trained checkpoints made available by Micallef et al. (2022) for the BERTu [13] and mBERTu [14] models and finetune them on the four downstream tasks as described in the methodology. We denote mBERTu as massively multilingual since it is mBERT further pre-trained on Maltese, where mBERT was originally pre-trained on 104 languages. Our distillation procedure combines only two related languages at a time into a single model and thus is denoted as 'Specific Multilingual' in the reported results.

Results and Discussion

The results across the four downstream tasks are reported in Table 8. We find that the monolingual model BERTu performs best on the semantic tasks and the multilingual model mBERTu performs the best on the token-level tasks.

In all the downstream tasks, we find the distilled models perform below the monolingual and massively multilingual models. However, on the NER task, the mergedistill-mt-en model with a mean Span F1 of 72.64 performs comparably to BERTu with 72.04.

Our student models use only 0.6% (15.3MB) of Maltese text to train compared to the 2.52GB of Maltese text used by BERTu and mBERTu and is also pre-trained only on 10% of the total tokens seen by BERTu as shown in Figure 25. Despite being pre-trained on less data and using less compute time, the student models still perform comparably well. The best-performing student model retains most of its performance of the best-performing model with 93.7% on News Tagging, 85.9% on Sentiment Analysis, 90.4% on NER, and 98.3% on POS Tagging. This provides evidence to suggest that the distillation-based pretraining approach is data efficient.

[13] `https://huggingface.co/MLRS/BERTu`

[14] `https://huggingface.co/MLRS/mBERTu`

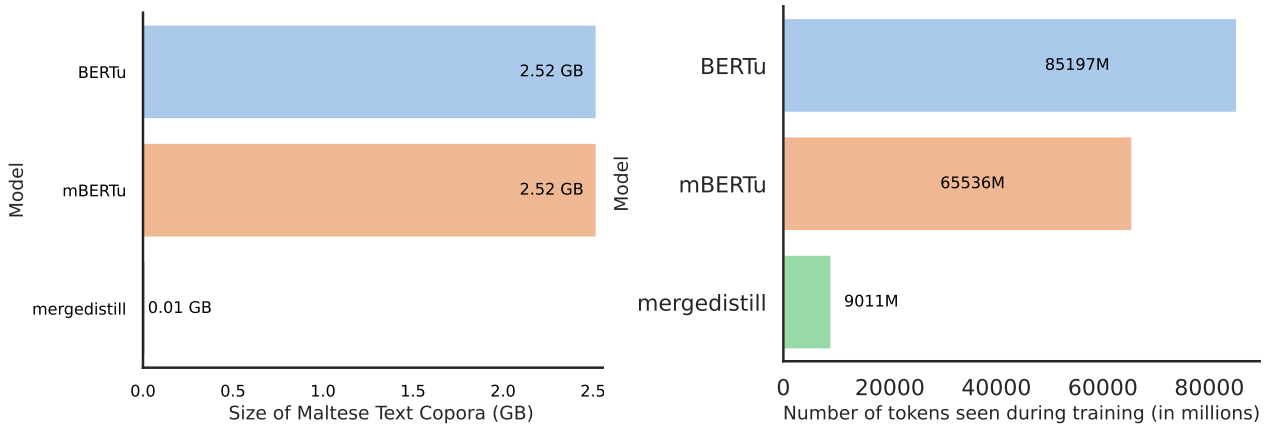

Figure 25: Comparison of the pre-training parameters for the models

Type	**Model**	**News Tagging** (Macro F1)	**Sentiment Analysis** (Accuracy)
Monolingual	BERTu	**67.04** ± 2.02	**85.96** ± 2.14
Massively Multilingual	mBERTu	66.65 ± 1.34	82.45 ± 2.48
Specific Multilingual	mergedistill-mt-it	62.64 ± 0.91	72.07 ± 1.87
	mergedistill-mt-ar	62.80 ± 1.99	73.87 ± 1.21
	mergedistill-mt-en	61.21 ± 1.33	69.29 ± 1.81

(a) Performance on semantic tasks

Type	**Model**	**NER** (Span F1)	**POS Tagging** (Accuracy)
Monolingual	BERTu	72.04 ± 5.92	97.86 ± 0.05
Massively Multilingual	mBERTu	**80.31** ± 2.20	**98.22** ± 0.11
Specific Multilingual	mergedistill-mt-it	68.33 ± 3.16	96.55 ± 0.08
	mergedistill-mt-ar	66.30 ± 3.29	96.16 ± 0.11
	mergedistill-mt-en	72.64 ± 3.94	96.41 ± 0.11

(b) Performance on syntactic/token-level tasks

Table 8: Experimental results, grouped by the downstream task.

8 Conclusion

8.1 Summary

In this master's thesis, we independently implement the MergeDistill framework within the context of a low-resource language Maltese.

We verify that knowledge distillation when pretraining a BERT-based model for Maltese from an existing Maltese monolingual model works and show that it improves performance on the downstream tasks while being data efficient.

Furthermore, we demonstrate that multi-teacher distillation, achieved by combining a monolingual Maltese model BERTu with teacher models from three languages related to Maltese – Italian, Arabic, and English has better downstream performance compared to single-teacher distillation only from BERTu. When compared with both monolingual and multilingual models for Maltese, we observe that our distilled student models, despite being pretrained on only 15.3MB of Maltese text for 10% of the pretraining steps, retain an average of 85% of their downstream task performance.

Our implementation is language-independent and easily extensible to new low-resource languages as well and the entire pipeline is available leveraging the popular Huggingface ecosystem of transformers and datasets libraries. We make available pre-processed versions of the Wikipedia corpus for Italian [15], Maltese [16], and Arabic [17] as well as datasets with masked tokens annotated with predictions from the respective teacher BERT-based models [18]. We also make available our pre-trained student models for Maltese-Italian[19], Maltese-Arabic [20], and Maltese-English [21] for future work.

[15] https://huggingface.co/datasets/amitness/wikipedia_it
[16] https://huggingface.co/datasets/amitness/wikipedia_mt
[17] https://huggingface.co/datasets/amitness/wikipedia_ar
[18] https://huggingface.co/amitness?search_datasets=logits-mt
[19] https://huggingface.co/amitness/mergedistill-mt-it-128-v2
[20] https://huggingface.co/amitness/mergedistill-mt-ar-128-v2
[21] https://huggingface.co/amitness/mergedistill-mt-en-128-v2

8.2 Future Work

In this section, we discuss the possible research areas for future work to build upon the work presented in this thesis.

In our study, we perform knowledge distillation using Maltese and one related language at a time (Maltese-Italian, Maltese-Arabic, and Maltese-English) as teachers. The results indicate that each combination performs well on some different downstream tasks. Therefore, it could be interesting to explore the impact of combining models from more than two languages simultaneously and assess how that affects downstream performance (e.g., Maltese-Italian-Arabic, Maltese-Italian-English, Maltese-Arabic-English, and Maltese-Italian-Arabic-English).

Additionally, we only pre-trained our models with a batch size of 128 for 250,000 steps due to computational constraints, while Khanuja et al. (2021) used a batch size of 4096. It can be interesting to study the impact of larger batch sizes in this framework. We hypothesize that a larger batch size would enhance the performance of the student model and bring it closer to the monolingual and multilingual models.

The teacher model for Arabic in our work uses a non-Latin script compared to Maltese which uses a Latin script. As such, there is limited overlap in their vocabulary. An interesting follow-up work could be to transliterate Maltese Wikipedia as well as the downstream tasks from Latin script to Arabic script following Micallef et al. (2023). Then a student model could be distilled from AraBERT on a mixture of Arabic and transliterated Maltese Wikipedia. This model can be evaluated on transliterated downstream tasks and compared against a regular AraBERT + BERTu student model. This could help us understand the impact of script when distilling from Arabic models.

Furthermore, in our work, we chose the same teacher models for English, Italian, and Arabic as those used in Khanuja et al. (2021) to be consistent. AraBERT was selected as the teacher model for Arabic. However, newer Arabic language models, such as CAMeLBERT (Inoue et al., 2021), have been released and could be considered as alternative teacher models. A comparative analysis could be carried out to assess the performance of CAMeLBERT as a teacher model against AraBERT.

In our study, we only used Maltese Wikipedia for the distillation process, but consid-

ering the small size of the Maltese Wikipedia, it might be worthwhile to investigate the use of an alternative dataset. This dataset should have the same number of sentences but come from various sources, like Korpus Malti (Micallef et al., 2022). This exploration could help us understand how the domain of the pretraining data affects the distillation process. Additionally, we can explore the potential of generating an increasingly larger amount of pretraining data for distillation using a corpus like Korpus Malti. This would allow us to determine the minimum amount of data needed to achieve performance comparable to that of a monolingual BERTu model pre-trained entirely from scratch on the entire Korpus Malti.

8.3 Revisiting the objectives

In this section, we review the objectives set at the beginning of the study and relate them to the results we achieved.

First, we wanted to evaluate the effectiveness of Knowledge Distillation to pre-train a student model by distilling from an existing monolingual BERT-based model for Maltese when provided a limited amount of text corpora. We find that the model pre-trained using distillation converged faster and also had better downstream performance.

Next, we were able to successfully implement the MergeDistill (Khanuja et al., 2021) framework to merge the Maltese monolingual model with each of the monolingual models from Arabic, Italian, and English and evaluate it on downstream tasks

We also compared the performance difference between distilling just from a single Maltese monolingual model vs distilling from multiple teachers and found that each of the related languages helped in the downstream tasks.

Finally, we compared the performance of our specific multilingual BERT model to that of the existing monolingual model BERTu and multilingual model mBERTU and showed that our model retains an average of 85% of the downstream task performance even when it has been trained on far smaller corpus and far less number of training steps. However, unlike Khanuja et al. (2021), we didn't find the specific multilingual models to be better than the monolingual models. This is expected given that we only train for around 4% of their total training steps and on a comparatively smaller batch size. All the

languages that Khanuja et al. (2021) used had comparatively large text corpora available in Wikipedia compared to our very low resource scenario.

9 Bibliography

References

Abdelali, A., Darwish, K., Durrani, N., and Mubarak, H. (2016). Farasa: A fast and furious segmenter for Arabic. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 11–16, San Diego, California. Association for Computational Linguistics.

Antoun, W., Baly, F., and Hajj, H. (2020). Arabert: Transformer-based model for arabic language understanding. *arXiv preprint arXiv:2003.00104*.

Brincat, J. (2011). *Maltese and other languages: A linguistic history of Malta*. Midsea Books, Malta.

Bucila, C., Caruana, R., and Niculescu-Mizil, A. (2006). Model compression. volume 2006, pages 535–541.

Cer, D., Diab, M., Agirre, E., Lopez-Gazpio, I., and Specia, L. (2017). SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada. Association for Computational Linguistics.

Chau, E. C., Lin, L. H., and Smith, N. A. (2020). Parsing with multilingual BERT, a small corpus, and a small treebank. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1324–1334, Online. Association for Computational Linguistics.

Clark, K., Khandelwal, U., Levy, O., and Manning, C. D. (2019a). What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy. Association for Computational Linguistics.

Clark, K., Luong, M.-T., Khandelwal, U., Manning, C. D., and Le, Q. V. (2019b). BAM! born-again multi-task networks for natural language understanding. In *Proceedings*

of the 57th Annual Meeting of the Association for Computational Linguistics, pages 5931–5937, Florence, Italy. Association for Computational Linguistics.

Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, É., Ott, M., Zettlemoyer, L., and Stoyanov, V. (2020). Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451.

Conneau, A., Rinott, R., Lample, G., Williams, A., Bowman, S., Schwenk, H., and Stoyanov, V. (2018). XNLI: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2475–2485, Brussels, Belgium. Association for Computational Linguistics.

Cortis, K. and Davis, B. (2019). A social opinion gold standard for the Malta government budget 2018. In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*, pages 364–369, Hong Kong, China. Association for Computational Linguistics.

de Vries, W., van Cranenburgh, A., Bisazza, A., Caselli, T., van Noord, G., and Nissim, M. (2019). Bertje: A dutch bert model.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (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.

Dingli, A. and Sant, N. J. (2016). Sentiment analysis on maltese using machine learning.

Dolan, W. B. and Brockett, C. (2005). Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

El-khair, I. A. (2016). 1.5 billion words arabic corpus. *arXiv preprint arXiv: Arxiv-1611.04033*.

Ganta, D. P., Gupta, H. D., and Sheng, V. S. (2021). Knowledge distillation via weighted ensemble of teaching assistants. In *2021 IEEE International Conference on Big Knowledge (ICBK)*. IEEE.

Gardner, M., Grus, J., Neumann, M., Tafjord, O., Dasigi, P., Liu, N. F., Peters, M., Schmitz, M., and Zettlemoyer, L. (2018). AllenNLP: A deep semantic natural language processing platform. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 1–6, Melbourne, Australia. Association for Computational Linguistics.

Gatt, A. and Čéplö, S. (2013). Digital Corpora and Other Electronic Resources for Maltese. In *Proceedings of the International Conference on Corpus Linguistics*, pages 96–97. UCREL, Lancaster, UK.

Gokaslan, A. and Cohen, V. (2019). Openwebtext corpus. `http://Skylion007.github.io/OpenWebTextCorpus`.

Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network.

Honnibal, M., Montani, I., Van Landeghem, S., and Boyd, A. (2020). spaCy: Industrial-strength Natural Language Processing in Python.

Inoue, G., Alhafni, B., Baimukan, N., Bouamor, H., and Habash, N. (2021). The interplay of variant, size, and task type in Arabic pre-trained language models. In *Proceedings of the Sixth Arabic Natural Language Processing Workshop*, pages 92–104, Kyiv, Ukraine (Virtual). Association for Computational Linguistics.

Jiao, X., Yin, Y., Shang, L., Jiang, X., Chen, X., Li, L., Wang, F., and Liu, Q. (2020). TinyBERT: Distilling BERT for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4163–4174, Online. Association for Computational Linguistics.

Khanuja, S., Johnson, M., and Talukdar, P. (2021). MergeDistill: Merging language models using pre-trained distillation. In *Findings of the Association for Computational*

Linguistics: ACL-IJCNLP 2021, pages 2874–2887, Online. Association for Computational Linguistics.

Lample, G. and Conneau, A. (2019). Cross-lingual language model pretraining.

Lhoest, Q., Villanova del Moral, A., Jernite, Y., Thakur, A., von Platen, P., Patil, S., Chaumond, J., Drame, M., Plu, J., Tunstall, L., Davison, J., Šaško, M., Chhablani, G., Malik, B., Brandeis, S., Le Scao, T., Sanh, V., Xu, C., Patry, N., McMillan-Major, A., Schmid, P., Gugger, S., Delangue, C., Matussière, T., Debut, L., Bekman, S., Cistac, P., Goehringer, T., Mustar, V., Lagunas, F., Rush, A., and Wolf, T. (2021). Datasets: A community library for natural language processing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 175–184, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach.

Loshchilov, I. and Hutter, F. (2019). Decoupled weight decay regularization.

Martínez-García, A., Badia, T., and Barnes, J. (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.

Micallef, K., Eryani, F., Habash, N., Bouamor, H., and Borg, C. (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.

Micallef, K., Gatt, A., Tanti, M., van der Plas, L., and Borg, C. (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.

Mohammadshahi, A., Nikoulina, V., Berard, A., Brun, C., Henderson, J., and Besacier, L. (2022). SMaLL-100: Introducing shallow multilingual machine translation model for low-resource languages. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8348–8359, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Muller, B., Anastasopoulos, A., Sagot, B., and Seddah, D. (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.

Nagel, S. (2016). Cc-news. `https://commoncrawl.org/blog/news-dataset-available`.

Nakayama, H. (2018). seqeval: A python framework for sequence labeling evaluation. Software available from https://github.com/chakki-works/seqeval.

Nozza, D., Bianchi, F., and Hovy, D. (2020). What the [mask]? making sense of language-specific bert models.

Ortiz Suárez, P. J., Sagot, B., and Romary, L. (2019). Asynchronous pipeline for processing huge corpora on medium to low resource infrastructures. In *Proceedings of the 7th Workshop on the Challenges in the Management of Large Corpora, CMLC-7*. Leibniz-Institut für Deutsche Sprache.

Pan, X., Zhang, B., May, J., Nothman, J., Knight, K., and Ji, H. (2017). Cross-lingual name tagging and linking for 282 languages. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1946–1958, Vancouver, Canada. Association for Computational Linguistics.

Park, D. and Ahn, C. W. (2019). Self-supervised contextual data augmentation for natural language processing. *Symmetry*, 11:1393.

Pfeiffer, J., Goyal, N., Lin, X., Li, X., Cross, J., Riedel, S., and Artetxe, M. (2022). Lifting the curse of multilinguality by pre-training modular transformers. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3479–3495, Seattle, United States. Association for Computational Linguistics.

Rosner, M. and Borg, C. (2022). *Report on the Maltese Language*. Language Technology Support of Europe's Languages in 2020/2021. Maria Giagkou, Stelios Piperidis, Georg Rehm, Jane Dunne (Series Editors). Available online at `https://european-language-equality.eu/deliverables/`.

Sanh, V., Debut, L., Chaumond, J., and Wolf, T. (2019). Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Schuster, M. and Nakajima, K. (2012). Japanese and korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5149–5152. IEEE.

Schweter, S. (2020). Italian bert and electra models.

Sennrich, R., Haddow, B., and Birch, A. (2016). Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., and Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Sun, S., Cheng, Y., Gan, Z., and Liu, J. (2019). Patient knowledge distillation for bert model compression.

Sun, Z., Yu, H., Song, X., Liu, R., Yang, Y., and Zhou, D. (2020). MobileBERT: a compact task-agnostic BERT for resource-limited devices. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2158–2170, Online. Association for Computational Linguistics.

Szymański, P. and Kajdanowicz, T. (2017). A scikit-based Python environment for performing multi-label classification. *ArXiv e-prints*.

Tang, R., Lu, Y., Liu, L., Mou, L., Vechtomova, O., and Lin, J. (2019). Distilling task-specific knowledge from bert into simple neural networks. *arXiv preprint arXiv:1903.12136*.

Tiedemann, J. (2012). Parallel data, tools and interfaces in opus. In Chair), N. C. C., Choukri, K., Declerck, T., Dogan, M. U., Maegaard, B., Mariani, J., Odijk, J., and Piperidis, S., editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey. European Language Resources Association (ELRA).

Tsai, H., Riesa, J., Johnson, M., Arivazhagan, N., Li, X., and Archer, A. (2019). Small and practical BERT models for sequence labeling. 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 3632–3636, Hong Kong, China. Association for Computational Linguistics.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. (2018). GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Warstadt, A., Singh, A., and Bowman, S. R. (2019). Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.

Wenzek, G., Lachaux, M.-A., Conneau, A., Chaudhary, V., Guzmán, F., Joulin, A., and

Grave, E. (2019). Ccnet: Extracting high quality monolingual datasets from web crawl data.

Williams, A., Nangia, N., and Bowman, S. (2018). A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T. L., Gugger, S., Drame, M., Lhoest, Q., and Rush, A. M. (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.

Wu, S. and Dredze, M. (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.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Łukasz Kaiser, Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M., and Dean, J. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation.

Yang, Z., Shou, L., Gong, M., Lin, W., and Jiang, D. (2019). Model compression with multi-task knowledge distillation for web-scale question answering system.

Yoo, A. B., Jette, M. A., and Grondona, M. (2003). Slurm: Simple linux utility for resource management. In Feitelson, D., Rudolph, L., and Schwiegelshohn, U., editors, *Job Scheduling Strategies for Parallel Processing*, pages 44–60, Berlin, Heidelberg. Springer Berlin Heidelberg.

Zeroual, I., Goldhahn, D., Eckart, T., and Lakhouaja, A. (2019). OSIAN: Open source international Arabic news corpus - preparation and integration into the CLARIN-infrastructure. In *Proceedings of the Fourth Arabic Natural Language Processing Workshop*, pages 175–182, Florence, Italy. Association for Computational Linguistics.

Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A., and Fidler, S. (2015). Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *2015 IEEE International Conference on Computer Vision (ICCV)*. IEEE.

10 Appendices

A.1 Creation of Semantic Benchmark Datasets

In this section, we describe how the two new semantic benchmark datasets were created in detail. The datasets were created by the author as part of an internship work outside the scope of this thesis.

Table 9 shows the size of the dataset for the two tasks for various splits as well as the evaluation metrics and the source domain from where the dataset was created.

Corpus	**|Train|**	**|Validation|**	**|Test|**	**Metric**	**Domain**
News Tagging	10.8k	2.3k	2.3k	Macro F1	news
News NLI	17.8k	3.81k	3.81k	Macro F1	news

Table 9: Statistics of the semantic benchmark tasks

We proposed two news tasks by repurposing a web corpus of Maltese text called Korpus Malti Micallef et al. (2022). The corpus was assembled by gathering text from various sources spanning different areas, including news, academic theses, Wikipedia, blogs, court documents, and more. We take one subset relevant to our work called *press_mt* which contains news articles along with their URL, title, article contents, and the associated tags.

We clean the news subset for our approach. First of all, we filter the dataset to only include article pages and remove pages such as category or author pages. We also normalize all the URLs to use a consistent format so that we can identify the news portal from the article URL. We drop duplicate articles from the dataset and any articles with no article content. We also drop articles whose article content is less than ten words.

During our analysis, we found that certain article contents contained CSS or JavaScript code when they were originally scraped to be included in Korpus Malti. We wrote specific regular expressions for each news portal to find those parts and removed them from the contents.

Figure 26: Automatic Dataset Generation Pipeline

We also find that certain news article contents include certain phrases repeated. For example, a portal Newsbook has the text "Read in English ." repeated in their article content. We remove such repeated phrases from the different portal contents.

With our final cleaned version of the news dataset, we propose the following two semantic tasks:

A.1.1 News Tagging (Task 1)

In this task, we build a multi-label news classification dataset to classify news article contents into a set of defined categories. We use a semi-automated pipeline with partial human annotation. We leverage the fact that news portals already assign specific tags on their articles and that can be used as our weak labels to define this task.

Korpus Malti (Micallef et al., 2022) already provides a corpus of 44823 articles scraped from various Maltese news portals. The raw corpus contains the article title, contents, the portal it was scraped from, and the tags used.

The categories used by the different news portals differ. We collected the 233 unique tags available in all the portals available. Then, two of our co-authors who are native speakers of Maltese manually merged similar tags into a higher-level category and defined

a list of tags that were too specific and could be dropped. This served as an annotation for the rest of the automated pipeline.

Some of the articles were only tagged as a general category in the portal (e.g. News, Local, Headlines, Uncategorised, Archived, etc.) and thus we dropped all those articles. This reduces our working dataset size from 44823 to 16058 articles. We also dropped tags that had less than 100 articles in total. With this, we get a total of 21 tags. We then calculate the co-occurrence between various categories and drop categories that overlap more than 75% with another category. This procedure removes four tags. At the end of our pipeline, we have 15,374 articles assigned to 17 categories.

We create 70% training, 15% validation, and 15% test splits using iterative stratification via the scikit-multilearn package (Szymański and Kajdanowicz, 2017).

A.1.2 News Entailment (Task 2)

We create a dataset for a binary task that checks if a statement in Maltese supports a given premise. We automatically generate this dataset using the news articles from the Korpus Malti.

We take news articles that have titles and descriptions from Korpus Malti and use the title as the premise and the description as the hypothesis. For 50% of the articles, we use the correct article title and description and assign a positive label. To generate negative samples, we tried three variants:

- **Random:** For an article, assign a random title from another article
- **Easy** For an article, get a random title from an article in a different cluster
- **Hard:** For an article, get a random title from an article in the same cluster

To assign clusters to the articles, we first vectorize the titles using TF-IDF and then apply singular value decomposition on the embeddings to generate a final 300-dimensional feature. KMeans algorithm was used to cluster the article titles. We determine the number of clusters by taking the square root of the total number of titles.