Modeling Second Language Acquisition with Pre-trained Neural Language Models

Álvaro J. Jiménez Palenzuela[a], Flavius Frasincar[a], Maria Mihaela Truşcă[b,*]

[a]*Erasmus University Rotterdam, P.O. Box 1738, 3000DR, Rotterdam, the Netherlands*
[b]*Bucharest University of Economic Studies, Piata Romana 6, 010374, Bucharest, Romania*

Abstract

Prediction of language mistakes is a task introduced by Duolingo as part of the Second Language Acquisition Modeling topic that aims to learn from the history of mistakes to improve the experience of language learners. Using transfer learning by means of pre-trained language models, we propose a framework that can learn the actual mistakes distribution according to which faraway words of a sentence have a higher chance to produce errors. To adapt the information provided by the pre-trained language models, more approaches based on feature extraction or fine-tuning were tried. However, according to our experiments, integrating these two options in a stack-and-finetune approach seems to be more appropriate for our task. Regarding the comparison of language models in terms of model distillation, we notice that distillation does not affect the effectiveness while significantly reducing the training time. We conclude that the model complexity should be adjusted to the specifics of the analysed problem and the distillation is an efficient option for low complexity corpora without considerably affecting the overall performance.

Keywords: Second Language Acquisition, Pre-trained Language model, Model Distillation, Fine-tuning, Feature extraction

1. Introduction

For many years, foreign languages have been taught in a traditional, non-data-driven way with little or no personalization of the contents. In contrast, language learning platforms such as Duolingo now collect data on millions of users that can be used to find patterns in

*Corresponding author.
Email addresses: alvarojimpal@gmail.com (Álvaro J. Jiménez Palenzuela), frasincar@ese.eur.nl (Flavius Frasincar), maria.trusca@csie.ase.ro (Maria Mihaela Truşcă)

language learning and make adjustments to the learning process. Given the vast number of people that study foreign languages, small improvements in the language learning process can have a huge impact

One of the key factors that influence success in second language acquisition is motivation (Gardner, 1972), which has been proved to be tightly related to contextualization, personalization, and provision of choices in a learning environment (Cordova and Lepper, 1996). Thus, it is expected that by tailoring the exercises, the language learners will remain more motivated, learn faster, and be less likely to drop out. To tailor the contents presented to each learner, we first need to gain insight into the language learning process. Being able to accurately determine whether a language learner has effectively learned a concept is a way of doing this.

Second Language Acquisition Modeling (SLAM) is a task that belongs to the educational data mining research domain (Ferreira-Mello et al., 2014; Mihaescu and Popescu, 2021). The main aim of the task is to predict the mistakes that language learners will make in the future, given their history of mistakes. This new research field was commenced recently by Duolingo with a challenge (Settles et al., 2018). Several teams submitted papers with their solutions to the problem of predicting the mistakes that certain language learners will make, given their learning history and some demographic information. The learning history is the sequence of sentences produced by a user, where each word is labeled with a zero or a one depending on whether the user wrote it correctly. Several approaches and techniques have been proposed in the past to model Second Language Acquisition (SLA); these approaches are reviewed in Section 2. However, there have been some major breakthroughs recently in the Deep Learning and Natural Language Processing (NLP) community (Young et al., 2018). In fact, Deep Learning and NLP are currently two very hot topics. Multi-head attention mechanisms and transfer learning through pre-trained language models have been key in the development of new state-of-the-art models (Ruder, 2019).

Transformer-based Language Models (Vaswani et al., 2017) have been a major breakthrough in the NLP field, achieving state-of-the-art results in popular benchmarks such as the General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2018) and its successor SuperGLUE (Wang et al., 2019a). These large language models have been proved to be successful across many traditional NLP tasks. In this paper, the focus is to

investigate how language models and transfer learning can also be applied to the SLAM task.

There are several signs that indicate that transfer learning through pre-trained language models might be useful for the SLAM task. First, word embeddings were the third most effective feature (after response time and days in course) according to the SLAM Task Overview paper (Settles et al., 2018). Second, the morpho-syntactic features were shown to not affect or even have weakly negative effects. In (Xu et al., 2018) and (Yuan, 2018) it is demonstrated that including these features hurt the performance of the models. This is counter-intuitive, as one would expect such linguistic features to be related to language acquisition. However, these features were generated with SyntaxNet (Andor et al., 2016), a language parser that was released prior to NLP's modern era of transformer-based models. As a result, there are many parsing errors easily observed when taking a quick glance at the dataset.

Previous models (see Section 2) have used contextualized word embeddings learned only with the small set of within-task sentences provided in the SLAM dataset (Settles, 2018). Alternatively, our model incorporates a pre-trained language model that generates much richer representations of the sentences to capture many more aspects of the language. Precisely, the input representations of our model are provided by Bidirectional Encoder Representations from Transformers (BERT) and its distilled version (DistillBERT). The contribution of our work can be summarized as follows:

- A state-of-the-art solution is proposed for the SLAM task that relies on transfer learning;
- Given the low complexity of the employed corpus, distillation turns out to boost the efficiency of our model compared to the conventional BERT model;
- The stack-and-finetune approach used to extract the information stored in a pre-trained language model was proved to be the most suitable for the SLAM task.

2. Related Literature

This section provides an overview of the techniques proposed for the 2018 Duolingo Shared Task on SLAM. We start by presenting the best performing models submitted to the 2018 Duolingo competition and then provide a summary of the most recent methods relevant to SLAM. A thorough review and meta-analysis of all the papers that were submitted to the

Duolingo competition are presented in (Settles et al., 2018). Despite not being the most effective approach, our solution has high-efficiency rates due to the use of DistillBERT. Therefore, the aim of our paper is twofold. Firstly, to introduce transfer learning in the field of SLAM. Secondly, to prove that distillation is an efficient alternative to the widely used BERT model with small performance reductions, especially when the complexity of the employed data is low.

Osika et al. (2018) used an ensemble model that combines the predictions of an Recurrent Neural Network (RNN) with Gradient Boosted Decision Trees (GBDT). They also engineered a set of additional features such as the number of times the current token has been practiced and the time since the current token was last seen. The authors excluded morphological features due to their poor predictive ability when evaluated by a decision tree model. The three most valuable features were *token*, *user*, and *format*. Furthermore, they observed that 4% of the instances with the least common words in the development set contribute to 10% of the prediction error.

Xu et al. (2018) designed an RNN with four types of encoders: a context encoder (both at word level and character level), a linguistic feature encoder (including part-of-speech, morphological, and syntactic information), a user information encoder (including user ID, country, and days in course), and a format information encoder (including device type and response time). The context encoder consists of two LSTMs that work at both word and character levels, respectively. The authors found out that the context and the format encoders are the most effective ones and that the linguistic encoder is the least effective.

Rich et al. (2018) used an ensemble of GBDTs with features motivated by theories from the psychology literature. They engineered features that aimed to capture the motivation and diligence of users. Other features such as corpus frequency and L1-L2 cognates were included. The authors concluded that morphological features and part-of-speech tags contributed very little to the predictive ability. Furthermore, they indicated that word order, subject-verb matching, and other grammatical rules are aspects in which users commonly make mistakes, and that explains the importance of considering the word contexts.

Kaneko et al. (2018) proposed a system with two components: a predictive Bi-LSTM that predicts whether a learner has made a mistake for the current word and a history LSTM that tracks the learning history of each specific learner. The output of the predictive Bi-LSTM

is fed into the history LSTM at each step. The authors trained a single model for all three language tracks (Spanish, English, and French) without any engineered features or language-specific information. An ablation experiment confirmed the importance of the history LSTM; the AUC decreased from 0.834 to 0.648 when excluding it.

Bestgen (2018) used a model based on logistic regression. Multiple conjunctive features (that is, features that are built by combining several primitive features) were engineered by taking word n-grams and combining them with metadata about the exercises and the users. The author decided not to incorporate morpho-syntactic information due to the lexical and syntactical simplicity of the exercises. The most effective conjunctive features included the tokens and the exercise format.

A more recent work proposed for SLAM is introduced in (Hu et al., 2020) by means of an encoder-decoder architecture. The encoding part is represented by two modules for capturing both the input sentences features and the meta-information (user and exercise features). To decode the encoded information, the method employs a multi-layer perceptron that yields the final token-level predictions. The proposed method is further enriched by a multi-task learning approach used to provide predictions for multiple language learning datasets simultaneously.

A similar encoder-decoder is also approached by Ruan et al. (2021). However, the meta-information is not further encoded, and the decoder has an auto-regressive nature to better capture the dependencies between words. In the end, the predictions are generated by a variational inference layer. To mimic the most effective method introduced for SLAM by Osika et al. (2018), Ruan et al. (2021) pack their solution as an ensemble model that combines the proposed encoder-decoder with a GBDT model.

Sense et al. (2021) are concerned to enhance a machine learning model via a cognitive approach. While given a sufficient volume of training data, the machine learning model is able to perform satisfactorily, training on limited data might benefit from the insights of a cognitive model. Considering the SLAM data on the English track as a reference, GBDT is selected as the main token classifier, and the Predicted Performance Equation (PPE) is chosen as the cognitive model. Despite the small performance margin reported between GBDT and PPE-GBDT models on the entire data, the margin is more significant for small subsets.

In addition to the above works that have as the main target the prediction of the token-level mistakes, there are some worth mentioning works that have slightly different aims for

improving the users' learning experience. The model proposed by Srivastava and Goodman (2021) has a coarser approach predicting the correct/incorrect label at the exercise level instead of the token level. The labels are simply assigned with respect to the presence of a right or wrong answer in the exercise solution. Similar to our approach, the proposed binary classifier relies on a transformer-based language model - GPT-2 (Radford et al., 2019). To improve the learning experience, the authors also include an exercise generator model that returns a new exercise based on the quality of the previous answers. Basically, more wrong/correct answers require the generation of simpler/more complex exercises. Next, according to Wu et al. (2020), the purpose of the Duolingo dataset to predict the users' capability to answer correctly is redefined to infer the proficiency of the language learners. The solution is provided by considering the Item Response Theory in an approach that relies on the principle of variational inference.

3. Methodology

3.1. Language Models

Language modeling is the task of predicting the word that will appear next in a given text. This simple idea has been shown to lead to very powerful representations of language. Language models are trained on huge corpora that contain unlabeled data, what is commonly referred to as unsupervised training. In this way, one can take advantage of the enormous amounts of text data that are available and learn how languages work.

Language models compute the probability of a sequence of words appearing in a certain sentence. This probability $P(w_1, \ldots, w_N)$ over the N words of the input sentence can be expressed as the product of the conditional probabilities $P\left(w_i | w_1, \ldots, w_{i-1}\right)$ of each word w_i. One might also choose a window of n previous words to compute the conditional probabilities, which is approximately equal:

$$P\left(w_1, \ldots, w_N\right) = \prod_{i=1}^{i=N} P\left(w_i | w_1, \ldots, w_{i-1}\right) \tag{1}$$

$$\approx \prod_{i=1}^{i=N} P\left(w_i | w_{i-(n-1)}, \ldots, w_{i-1}\right) \tag{2}$$

Neural language models, also referred to as continuous space language models, make use of neural networks to solve the task of predicting the next word in a sentence. As opposed to tra-

ditional language models, neural language models represent words as non-linear combinations of weights in a neural network, in a so-called distributed way. While the first neural language models used feed-forward and recurrent networks, the latest rely heavily on attention-based mechanisms.

Several neural language models have been conceived in the last years, achieving SoTA results across many NLP tasks. In fact, the emergence of these models was described by Sebastian Ruder as "the ImageNet[1] moment of NLP". The latest neural language models such as ULMFit (Howard and Ruder, 2018), GPT-2 (Radford et al., 2019), and BERT (Devlin et al., 2019) are able to capture many more complex language phenomena (e.g., polysemy and negation). As opposed to using word embeddings, which are very limited representations of language, language models allow us to pretrain an entire model and not just the first layer. Hence, instead of training a language model from scratch, one can take advantage of transfer learning by taking a language model that has been pre-trained on huge corpora (usually millions of sentences) and fine-tuning it to solve a specific task. In this way, complex language phenomena do not have to be learned from scratch every time a model is trained for a certain task. As a result, one needs significantly fewer tagged examples and much less computing power to train a model as compared to using word embeddings. This is of great importance, since collecting tagged data can be very expensive or even unfeasible in the case of minority languages (e.g., Estonian) or for certain tasks (e.g., sarcasm detection).

3.2. Transformer Architecture

The transformer architecture was proposed by Vaswani et al. (2017), and it is based only on attention mechanisms instead of RNNs. It consists of two parts: a stack of encoders and a stack of decoders. Both the encoders and the decoders are composed of modules that consist mainly of multi-head attention and feed-forward layers. All the encoders share the same structure; however, they do not share weights. As opposed to RNN-based encoder-decoder architectures, transformers are parallelizable and require less time to train.

Figure 1 displays a high-level schema of the transformer architecture. First, the inputs and outputs of the transformer are embedded into an n-dimensional space. In order to pre-

[1]The *ImageNet challenge* fostered the creation of computer vision models which achieved astonishing accuracy results through deep learning and transfer learning.

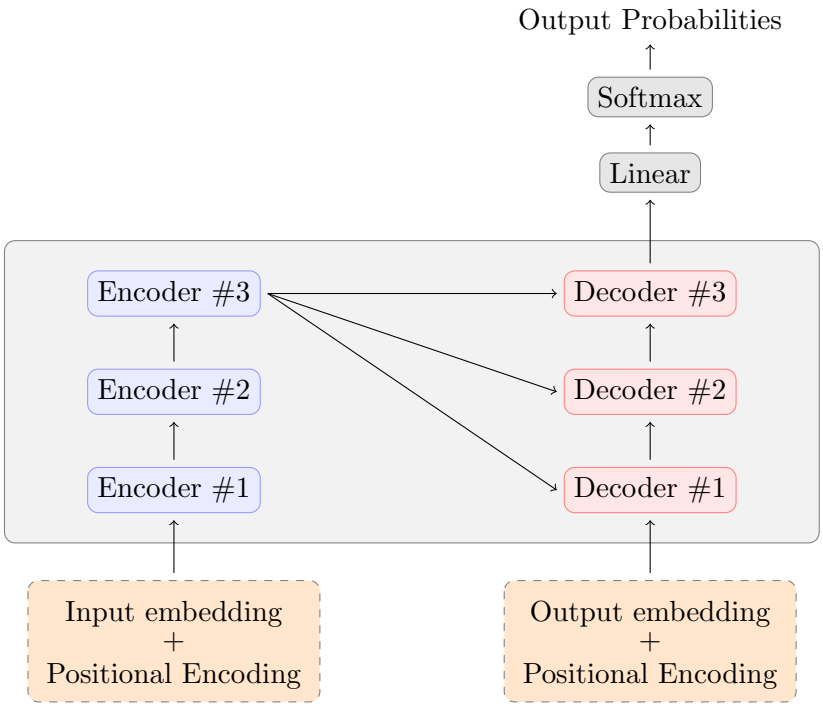
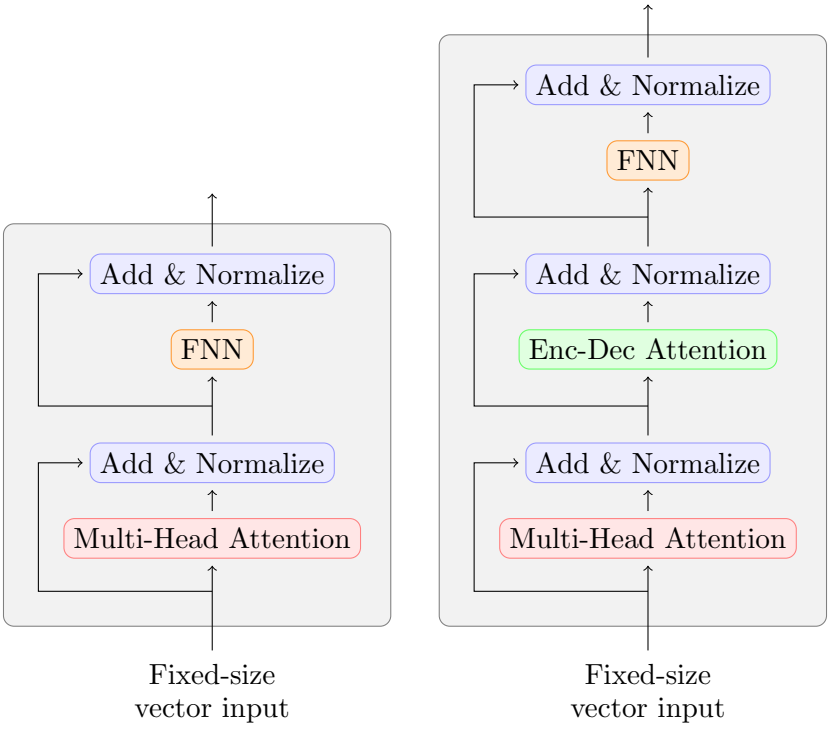


Figure 1: The simplified architecture of the transformer with three encoders and three decoders.

serve the sequential nature of the elements in the sequences (i.e., their relative positions), a positional encoding technique is used. These positions are added to the n-dimensional vector embeddings. These resulting word embeddings are then fed to the first encoder. The rest of the encoders have as input the output of the previous encoder, i.e., a fixed-length vector.

Each embedding flows through its own path in the encoder, making transformers parallelizable. Each encoder consists mainly of two sub-layers, as Figure 2(a) shows. First, the encoder's inputs flow through a self-attention layer. In this way, the encoder attends to other words in the input sentence in order to encode each word. Then, the outputs of the self-attention layer flow through a feed-forward neural network. Decoders (see Figure 2(b)) are similar to encoders; they also have a self-attention layer and a feed-forward neural network. However, between the two layers, the decoder has an additional attention layer for capturing relevant parts of the input sentence. Residual connections are used around each sub-layers of both the encoders and the decoders, followed by layer normalization.

The output of the decoder stack is fed to a fully connected neural network which projects it into a logits vector, i.e., a vector of size *vocab_size* where a score is assigned to each word in



(a) Encoder block (b) Decoder block

Figure 2: A more detailed architecture of the transformer including separate blocks for the Encoder and Decoder.

the vocabulary. Then, a softmax layer converts these scores into probabilities, and the word with the highest probability is selected as the output of the transformer.

3.3. BERT

In late 2018, Devlin et al. released BERT (Bidirectional Encoder Representations from Transformers), a language representation model based on the popular transformer architecture widely used for different NLP tasks that vary from sentiment classification (Meškelė and Frasincar, 2020) to multi-term response selection (Li et al., 2021). Unlike previous models such as ULMFiT (Howard and Ruder, 2018) and GPT (Radford et al., 2018), BERT learns deep bidirectional representations by taking into account both the left and right context of a word when constructing its representation. Devlin et al. show that by adding only one additional layer to BERT and fine-tuning it, state-of-the-art results can be achieved for a large variety of sentence-level and token-level NLP tasks, outperforming many task-specific architectures.

BERT only uses encoders in its architecture and comes in two sizes: *base* and *large*. The base model consists of 12 encoder layers, 12 self-attention heads, hidden size 768, and a total of 110M parameters. The large model has 24 encoder layers, 16 self-attention heads, a hidden size of 1024, and a total of 340M parameters. Two datasets were used to pre-train BERT: BooksCorpus (800M words) (Zhu et al., 2015) and English Wikipedia (2,500M words). In this way, BERT learns rich language representations through unsupervised learning.

There are two ways to adapt BERT's pre-trained language representations to downstream tasks: *feature extraction* and *fine-tuning*. Feature extraction requires less computation time, as the model's layers are kept frozen (i.e., their weights are not updated), and the resulting vector embeddings can be reused. On the other hand, fine-tuning is more computationally intensive but leads to better results in many cases. Peters et al. (2019) explore these two adaptation strategies across different NLP tasks and conclude that their relative performance depends on the similarity of the pretraining and target tasks.

Two tasks were used to pre-train BERT: Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). MLM randomly masks 15% of the input tokens and predicts the masked tokens based only on their context. A [MASK] tag is assigned to each randomly masked input token. NSP uses pairs of sentences $(sentenceA, sentenceB)$ to capture the relationships between consecutive sentences. To that end, 50% of the training pairs are contiguous sentences selected from the text corpus and assigned the label $IsNext$, while the other half are non-contiguous sentences that are randomly selected and assigned the label $NotNext$.

BERT's input representation is the sum of three embeddings: *token*, *segment*, and *position* embeddings. The token embeddings are obtained from an embedding dictionary that contains 30,000 tokens, the segment embeddings allow BERT to distinguish between sentence A and sentence B, and the position embeddings are used to model the sequential nature of the words in the sentences. BERT uses two special tags, namely a [CLS] tag and a [SEP] tag. The [CLS] tag corresponds to the first input token and is used in order to construct an aggregate sequence representation. The [SEP] tag is used to separate sentences A and B. These tags are added in the preprocessing stage of the input sentences.

3.4. Model Distillation

Much of the attention in the NLP field has been paid to large transformer models since the release of BERT. However, at least some of the focus has shifted lately into making these massive models smaller in size and reducing their inference times. In this way, language models are becoming more accessible and manageable to researchers that lack huge computational power. There are several techniques available to reduce the size of a model. Knowledge distillation is a compression technique that allows us to obtain a reduced model, called the student, from a larger model, called the teacher. The student model is trained to mimic the teacher model's behavior. Other commonly used techniques are quantization and pruning. Quantization reduces the size of a model by decreasing the numerical precision of its weights, whereas pruning consists in removing parts of a model to reduce its size. This can be done in several ways; one can prune weights, neurons, or even weight matrices, e.g., by removing entire attention heads from transformers.

Model compression was introduced by Bucilă and Niculescu-Mizil (2006), who showed that complex ensembles of hundreds or thousands of base-level classifiers can be compressed into smaller, faster models with little loss in performance. A few years later, Hinton et al. (2015) developed this approach further and generalized it by using a different compression technique. Neural networks are usually trained to predict class probabilities by using a softmax layer that converts logits z_i into probabilities p_i. Hinton et al. introduce a temperature parameter in these softmax probabilities:

$$p_i = \frac{exp(z_i/T)}{\sum_j exp(z_j/T)}, \tag{3}$$

where T is a temperature parameter that controls the smoothness of the output distribution. Larger values result in a softer probability distribution over classes. That is, the temperature value T allows us to control the uncertainty in the teacher's output. The same temperature value is assigned to both the student and the teacher networks at training time. However, during inference T is set to 1 in order to recover a standard softmax.

As opposed to the usual classification problems in supervised learning where only the estimated probability of gold labels is maximized, knowledge distillation aims to train a smaller network that mimics the whole distribution of the output probability of the teacher network.

This is important because we are usually not only interested in the gold labels but also in other labels that might have a lower probability. For example, *book* might be the most likely masked word in the sentence `"This is an interesting [MASK]"`, but the probabilities assigned to the words *movie*, *story*, and *task* are also valuable and reflect part of the knowledge learned by the teacher network. To this end, cross-entropy over soft targets is used rather than over hard targets. In this way, the model learns to generalize better and faster. This is achieved by using a distillation loss $L_{ce} = \sum_i t_i * \log(s_i)$, where t_i is the estimated weight of the soft target s_i.

DistilBERT (Sanh et al., 2019) was trained on the same corpus as the original BERT model following the knowledge distillation approach proposed by Hinton et al. (2015), which was presented above. It retains 97% of BERT's performance on the GLUE benchmark and is 60% faster, while only having roughly half of BERT's parameters. Furthermore, it is the fifth best language model for the Semantic Textual Similarity task. In order to achieve this, Sanh et al. (2019) introduce a triple loss function that combines a masked language modeling loss L_{mlm}, a distillation loss L_{ce}, and a cosine-distance loss L_{cos} which intends to preserve the similarity of the vectors (embeddings) belonging to the teacher and the student. DistilBERT has the same general architecture as BERT, although the number of layers is reduced by half. However, the hidden size dimension is kept at 768, since most of the operations used in the transformer architecture are highly optimized and reducing it has a relatively smaller impact on the computational efficiency. DistilBERT is initialized with BERT's weights by taking one layer out of two, thus benefiting from their common dimensionality.

3.5. Model Architecture

The proposed model consists of three main parts: a set of embedding layers, a pre-trained Language Model (BERT or DistilBERT), and a set of layers on top of both. Since it has been shown that BERT-*base* (see Section 3.3) outperforms BERT-*large* in some cases (Goldberg, 2019), the proposed model is developed on the simpler BERT-base variant. As the number of observations available in our dataset is limited, and the model capacity of BERT-*large* is large, our model would likely overfit. Furthermore, in order to use BERT-*large* a larger GPU is required as well as longer training times. Figure 3 shows the architecture of our model as well as its inputs and outputs. The reported experimental results consider both BERT and

DistilBERT. The latter is a compact language model learned from BERT (see Section 3.4). The main advantage of DistilBERT is that it requires less computing resources while losing very little performance. Furthermore, it was trained without BERT's NSP task, which seems irrelevant for our task.

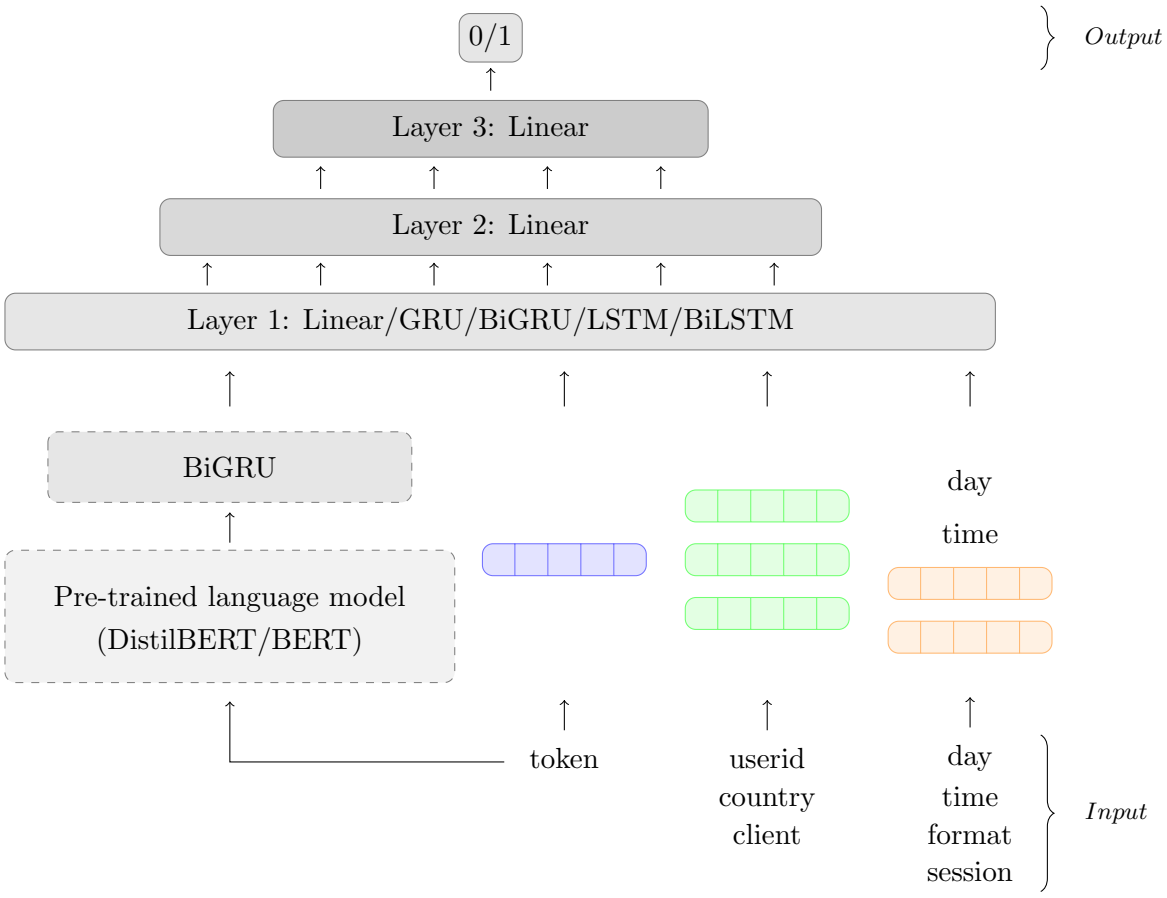


Figure 3: Model architecture.

The pre-trained language model (DistilBERT or BERT) takes as input a token belonging to an aligned reference answer. Hence, it is not required to learn an embedding from scratch for the pre-trained language model; there is already a pre-trained embedding layer in DistilBERT and BERT. The output of the pre-trained language model (of size 768 for both BERT and DistilBERT) is then fed into a bidirectional Gated Recurrent Unit (BiGRU) of size 256 in order to reduce the dimensionality of the pre-trained language model's output. Our model also takes as input several features that are provided in the dataset (see Section 4). An embedding is learned from scratch for each of these categorical features. All embedding layers

are of size 64 except for that of *userid*, which is of size 128, and *token*, which is of size 256. Those features have larger embeddings because of their higher cardinality (i.e., more unique values). Note that the token feature is input to both the pre-trained language model and an embedding layer. This allows our model to learn task-specific information more easily by means of a learned embedding and results in a more fair comparison. The continuous features (*day* and *time*) are fed into Layer 1 directly after standardizing their values. We also cap the *time* values at 100 seconds, as some of the values reported in the dataset are much above that. The output of the BiGRU on top of the pre-trained language model, the embeddings, and the continuous features are concatenated before being input into the upper layers of our model.

There are three layers on top of the pre-trained language model and the embedding layers. For Layer 1, several neural network architectures (linear and RNN layers) were tried with different configurations (number of layers and number of neurons). The next two layers are linear. The output of Layer 3 is a value in the range $[0.0, 1.0]$ predicting the probability that the user made a mistake at that token. A softmax activation function and a cross-entropy classification loss are used to obtain this probability.

The path through the entire network is as follows. First, the categorical features (*token*, *userid*, *country*, *client*, *format*, and *session*) are fed through their embedding layers. The token is also fed into the pre-trained language model, and its output goes through a BiGRU of size 256 with a dropout rate of 0.3, as a way of regularizing the input of Layer 1. Then, the embeddings, as well as the output from the BiGRU and the continuous features (*day* and *time*) are concatenated and fed into Layer 1, which can be a linear layer, a (Bi)GRU, or a (Bi)LSTM. The output of Layer 1 (of size 512) is fed into Layer 2, and that of Layer 2 (of size 32) into Layer 3 (of size 1), subsequently. A dropout rate of 0.5 is applied for Layers 1 and 2, which was found to be optimal for a wide range of neural networks and tasks (Srivastava et al., 2014). Note that the embedding layers and Layers 1, 2, and 3 are trained from scratch in every experiment reported in Section 6. Finally, a softmax layer is used to output a per-token probability of a mistake.

We use the PyTorch-Transformers library from HuggingFace (Wolf et al., 2019), a popular library with PyTorch implementations of state-of-the-art pre-trained language models. The models were trained on a 16GB Tesla P100 GPU provided by Google Colab. Our code is pub-

licly available at `https://github.com/alvaro768/slam-pre-trained-LM/`. The employed dataset is available at `https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/8SWHNO`.

The optimization was done by the Adam algorithm by means of the minimization of the binary cross-entropy between the predicted labels and the true labels given in Equation 4. The learning rate and the batch size are set to 3e-4 and 256, respectively. The max sequence length is 20, which is sufficient to accommodate any tokenized sentence (exercise instance) from our corpus.

$$L = -\frac{1}{T}\sum_{t=1}^{T}\left[y_t \cdot \log x_t + (1 - y_t) \cdot \log(1 - x_t)\right], \tag{4}$$

where x_t and y_t are the token and its label at time step t, respectively.

4. Data

The corpus of language learner data that we employ in this work was released by Duolingo to support the 2018 Duolingo Shared Task on Second Language Acquisition Modeling (Settles et al., 2018). The corpus collects data on three language tracks: English from Spanish (2.6k users), Spanish from English (2.6k users), and French from English (1.2k users). The task proposed by Duolingo consists in predicting the word-level mistakes that users will make given their learning history and some additional metadata such as the exercise format and the response time. In the following, L1 denotes the source language of a learner (not necessarily her native language) and L2 the target language (i.e., the language she is learning).

All the data available correspond to three types of exercises linked to written production: reverse translate, reverse tap, and listening. *Reverse translate* requires the user to translate from L1 to L2, *reverse tap* involves translating a sentence from L1 to L2 by selectively tapping words that are provided to the user, and the *listening* exercise consists in transcribing an L2 utterance. The three types of exercises that are provided involve active recall, which is a principle of efficient learning. Figure 4 illustrates these three types of exercises with examples.

The dataset includes all the sentences (exercise instances) produced by the users during the first 30 days of learning a language. Each exercise instance in the dataset collects information related to the exercise and the user. Figure 5 shows all the information available for a sample

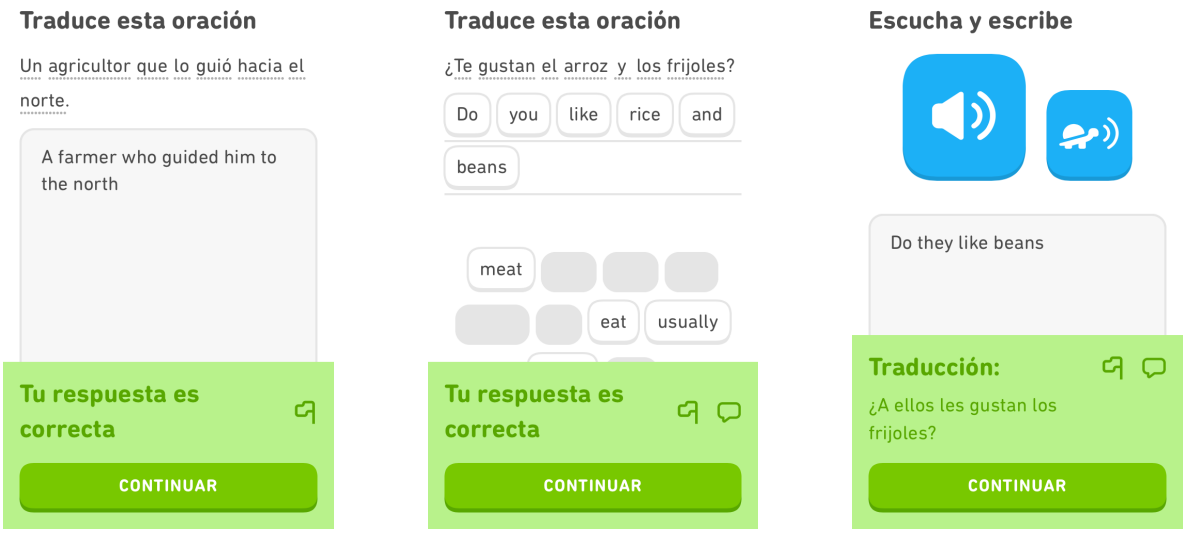


Figure 4: Types of exercises available on Duolingo: Reverse translate, Reverse tap, and Listening.

exercise instance. The following meta-data information are available: an anonymous *user ID*, the *countries* from which the user has done exercises, the number of *days* since the user started learning the language on Duolingo, the *client* (platform) which the learner used, the *session* type (lesson or practice), the exercise *format* (reverse translate, reverse tap, or listening), and the *time* (in seconds) that the learner took to finish the exercise. This meta-data is provided in the first line of each exercise instance (see Fig. 5). The rest of the lines include a token ID, the actual token (word) from the reference answer, and morpho-syntactic features (part of speech, morphology features, dependency parse labels, and dependency edges). Importantly, these features were not handcrafted but generated with SyntaxNet (Andor et al., 2016). The last column contains the labels: a 1 if the user made a mistake and a 0 otherwise.

In our work, following the approach proposed by Xu et al. (2018), there are three groups of features: token, user, and format features. Table 1 displays the grouping of these features. The morpho-syntactic features (part of speech, morphology features, dependency parse labels, and dependency edges) are excluded, as they were found to have weakly negative effects (Settles et al., 2018).

Note that the sentences provided in the corpus are not the actual responses that the learners submitted. Instead, their responses were aligned with the most similar reference (correct) answer and labels were assigned per-token: 0 if the learner's answer coincided with the most similar reference answer, and 1 otherwise. This matching was done by means of

#user:XUaq7Hc4 countries:NZ days:16.540 client:android session:practice format:listen time:17

Token ID	*Token*	*P.O.S.*	*Morphology feats.*	*Labels*	*Edges*	*Tag*
rx42Rld/0201	Yo	PRON	Case=Nom|Number=Sing|Person=1 PronType=Prs|fPOS=PRON++	nsubj	2	0
rx42Rld/0202	veo	VERB	Mood=Ind|Number=Sing|Person=1 Tense=Pres|VerbForm=Fin|fPOS=VERB++	ROOT	0	0
rx42Rld/0203	que	SCONJ	fPOS=SCONJ++	mark	4	0
rx42Rld/0204	tienes	VERB	Mood=Ind|Number=Sing|Person=2 Tense=Pres|VerbForm=Fin|fPOS=VERB++	ccomp	2	1
rx42Rld/0205	una	DET	Definite=Ind|Gender=Fem|Number=Sing PronType=Art|fPOS=DET++	det	6	0
rx42Rld/0206	gata	NOUN	Gender=Fem|Number=Sing|fPOS=NOUN++	dobj	4	0

Figure 5: Sample exercise instance (meta-information is provided on the first line; the rest of the lines offers information about the exercise).

Token	User	Format
token	userid	day
	country	time
	client	format
		session

Table 1: Feature grouping.

the finite-state transducer method (Mohri, 1997). Figure 6 illustrates this alignment with an example. The *learner* sentence is the sentence provided by the learner (possibly with mistakes) and the *reference* sentence is the most similar correct answer. Note that special characters such as accents and punctuation marks are not taken into account.

5. Evaluation

All datasets (for the three language tracks) are provided pre-partitioned; the first 80% of the learning data for each user is used for training, the next 10% for development, and the last 10% for testing. The performance of this per-token binary classification task is assessed by measuring the AUROC and the F_1-score, two evaluation metrics that are widely used for

learner:		Cu_antas	mans_anas	tienes		?
reference:	¿	Cuántas	manz_anas	tienes	tú	?
label:		0	1	0	1	

Figure 6: Sentence alignment example with the learners' responses allocated per token (labels indicate whether the learners' answers are correct).

classification problems. The AUROC measures the area under the ROC (Receiver Operating Characteristics) curve, which is a probability curve obtained by plotting TPR (True Positive Rate) against FPR (False Positive Rate). The F_1-scores were computed using a threshold of 0.5.

Precision is defined as the number of true positive results divided by the sum of true positives and false positives, and *recall* is the number of true positives divided by the sum of true positives and false negatives.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{5}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{6}$$

The F_1-score is the harmonic mean of precision and recall:

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{7}$$

6. Results

This section presents the performance evaluation of several models previously described in Section 3. These models incorporate the *token* feature, the user features *user id*, *country*, and *client*, and the format features *day*, *time*, *format*, and *session*. First, a baseline model is defined by including a linear layer as Layer 1 without a pre-trained language model. Then, different architectures are explored by comparing several types of layers and numbers of neurons for Layer 1. Finally, it is presented a comparison between several approaches for the incorporating of DistilBERT and BERT in our model. The considered approaches are: feature extraction (Peters et al., 2019), fine-tuning (Peters et al., 2019), and stack-and-finetune (Wang et al., 2019b).

6.1. Baseline Models

Table 2 presents the results of the baseline models. Settles et al. (2018) provide a baseline that uses a simple L2-regularized logistic regression trained via stochastic gradient descent (SGD) with all the dataset features provided. This logistic regression was trained using only the training set. This baseline model achieves an AUC value of 0.774 and an F_1 score of 0.190.

Our baseline model uses the neural architecture presented in Section 3. This baseline model features a 64-neuron linear layer as Layer 1. The rest of the model is left unchanged. Furthermore, it does not include a pre-trained language model. We also train this model using only the training set and report the results on the development set. Our baseline model achieves an AUC of 0.811 and an F_1 score of 0.379.

Model	AUC	F_1
SLAM baseline (Logistic Regression)	0.774	0.190
Our baseline (Neural Network)	0.811	0.379

Table 2: Baseline models results.

6.2. Architecture

Several architectural choices for Layer 1 are evaluated (see Fig. 3) before incorporating a pre-trained language model into our model. Table 3 reports the results for different combinations of layers (linear, GRU, BiLSTM, and BiGRU) and numbers of neurons (128, 256, and 512). Note that the bidirectional layers actually have twice as many neurons (e.g., a 128-neuron BiGRU has 256 neurons in total).

Using an RNN decoding layer (GRU, BiGRU, or BiLSTM) or larger layers (i.e., layers with more neurons) results in better AUC and F_1 scores. However, it can be clearly seen that the layer size has a much lower impact than the layer type. Furthermore, we observe that increasing the number of neurons exhibits diminishing returns.

Finally, it is also worth noting that biGRUs perform much better than their unidirectional counterpart. This confirms the importance of having bidirectional layers for text data. The difference between the BiGRU and the BiLSTM layers is small, with the former achieving slightly better AUC scores (our primary evaluation metric) but worse F_1 scores. As it results

in better AUC scores than the bidirectional LSTM layer, a 256-neuron BiGRU layer is used for our base model (see highlighted cells in Table 3). Furthermore, GRU layers are simpler (two gates instead of the three LSTM layers) and are computationally more efficient.

# of neurons	64		128		256		512	
Architecture	AUC	F_1	AUC	F_1	AUC	F_1	AUC	F_1
Linear layer	0.8176	0.3627	0.8207	0.3626	0.8219	0.3710	0.8223	0.3876
GRU layer	0.8345	0.4289	0.8352	0.4163	0.8367	0.4523	0.8391	0.4546
BiGRU layer	**0.8429**	0.4328	**0.8454**	0.4493	**0.8459**	0.4380	**0.8462**	0.4467
LSTM layer	0.8351	0.4199	0.8357	0.4214	0.8374	0.4481	0.8376	0.4480
BiLSTM layer	0.8425	**0.4563**	0.8445	**0.4650**	0.8448	**0.4662**	0.8452	**0.4603**

Table 3: Model performance for different architectures.

In order to determine the relative importance of the model's features before incorporating a pre-trained language model, an ablation study is performed. Table 4 shows the decrease in the AUC and F_1 scores after removing each set of features from our base model as described in Section 3.2. Leaving out the token feature has the largest impact on the model performance, resulting in a 0.1005 absolute decrease of the AUC score. This decrease is larger than that of leaving out both the format and the user features, which include seven categorical and numerical variables.

Ablated features	Δ AUC	Δ F_1
Token feature	**-0.1005**	**-0.2595**
Format features	-0.0498	-0.1669
User features	-0.0229	-0.1580

Table 4: Ablation study.

6.3. Distribution of Mistakes and Confusion Graphs

Next, the distribution of the actual mistakes made by the language learners in a sentence is compared with the distribution of the mistakes that were predicted by our model. The distribution plots in Figure 7 resembles a truncated right-skewed Normal distribution; users

make more mistakes at the beginning of a sentence. However, this does not take into account the fact that long sentences are less common in the dataset than short ones. Figure 8 controls for this and indicate the percentage of mistakes for each position in a sentence. These plots show that users are more likely to make a mistake in a word the further away a word is in a sentence. The reported numbers and percentages are computed per position in the sentence.

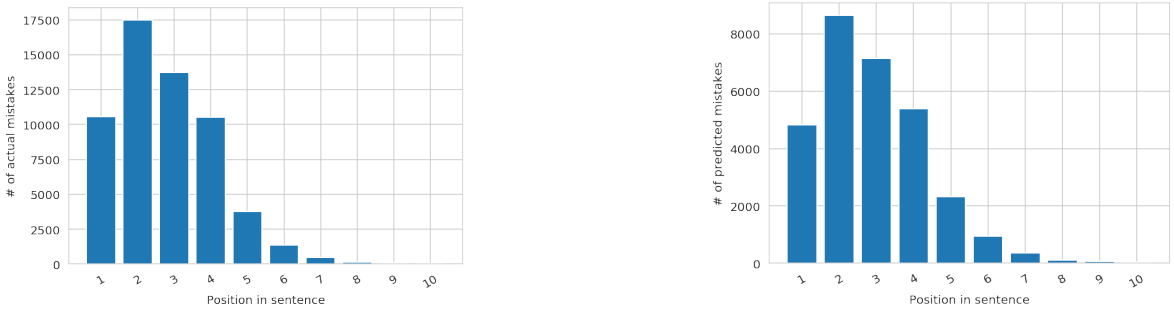


Figure 7: Corrected actual and predicted mistakes distribution.

The performance of our model at a token level is analysed hereinafter in order to understand when our model makes accurate predictions and when it fails to do so. Figure 9 shows two confusion graphs with the percentage of true positives, false positives, false negatives, and true negatives for every position in a sentence. The percentage of true positives (predicting a mistake and being correct) increase the further away a token is, while the percentage of true negatives (predicting no mistake and being correct) decreases. This is in line with the distribution of actual mistakes shown in Figure 8. Furthermore, the percentage of false positives (predicting a mistake and not being correct) and false negatives (predicting no mistake and not being correct) increase the further away a token is, which implies that the model makes more mistakes when less data is available (longer sentences are rarer in the dataset).

6.4. Feature Extraction

Feature extraction is the first approach used to incorporate a pre-trained language model in our solution. Precisely, we freeze all the layers of the pre-trained language model and extract the weight values of a subset of layers. The rest of the model (embedding layers and layers 1, 2, and 3) is trained after randomly initializing their weights. To that end, three models are built using (a) layers 1-2, (b) layers 3-4, and (c) layers 5-6 of DistilBERT, and three models using (a) layers 1-4, (b) layers 5-8, and (c) layers 9-12 of BERT. The outputs of these layers

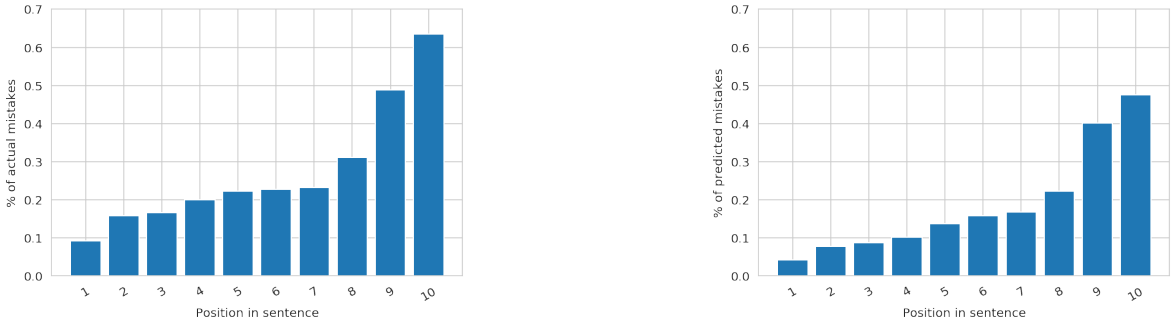


Figure 8: Actual and predicted mistakes distribution.

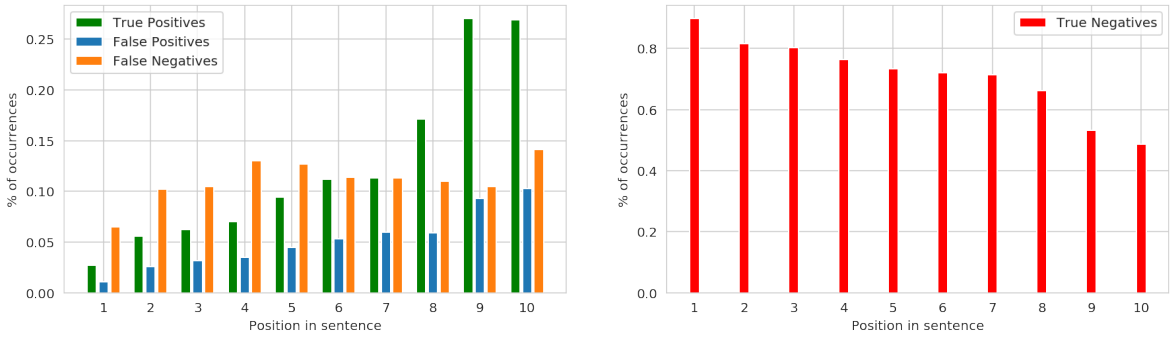


Figure 9: Confusion graphs.

are concatenated into a vector (of lengths 1,536 (2×768) and 3,072 (4×768), respectively) and then fed into a bidirectional GRU layer (see Figure 3). The effect of the token embedding is analysed by generating the results of the model with and without this embedding (see the top part of the table). These two models use the final output of (Distil)BERT.

As shown in Table 5, the AUC scores are higher than those achieved with our base model (without a pre-trained language model). Furthermore, the training times of the model with BERT are approximately twice as long as those of the model with DistilBERT. The times are given in *hh:mm:ss* format. The results of the models with and without a token embedding indicate that it is useful to have a token embedding as well as a pre-trained language model, which supports our decision of including both in our model. In this way, our model can learn SLAM-specific information that cannot be found in pre-trained models. As it has been shown that different layers of BERT learn different aspects of language, we compare the results across them. Extracting upper layers resulted in a lower AUC score for both DistilBERT and BERT; these layers capture linguistic phenomena that are more specific to the pre-training

Model	AUC	F_1	Time
NN+DistilBERT(with emb.)	0.8504	0.4647	1:26:54
NN+DistilBERT(no emb.)	0.8492	0.4693	1:54:49
NN+BERT(with emb.)	0.8501	0.4674	2:23:44
NN+BERT(no emb.)	0.8488	0.4548	3:07:24
NN+DistilBERT(extr. 1-2)	0.8512	**0.4772**	1:19:21
NN+DistilBERT(extr. 3-4)	0.8515	0.4591	1:19:07
NN+DistilBERT(extr. 5-6)	0.8508	0.4605	1:20:54
NN+BERT(extr. 1-4)	**0.8518**	0.4695	2:48:43
NN+BERT(extr. 5-8)	0.8503	0.4613	2:47:38
NN+BERT(extr. 9-12)	0.8495	0.4572	2:51:47

Table 5: Feature extraction results.

task it was trained on, while lower layers capture more general patterns. Last, note that NN+BERT(extr. 9-12) (i.e., concatenation of the features extracted from the four last layers) was the best-performing feature extraction approach in Devlin et al. (2019) for the Named Entity Recognition task. Given that our corpus is much different (and more simple) than the corpora on which BERT was trained, the lower layers of (Distil)BERT contain probably more useful information for SLAM than its upper layers.

6.5. Fine-Tuning

The fine-tuning approach, which consists in freezing some layers of the language model and updating (fine-tuning) the weights of the rest is explored next. The rest of the model (as depicted in Figure 3) is trained from scratch after randomly initializing the weights. Four different options are considered: fine-tuning only the last layer (layer 6), fine-tuning the last two layers (layers 5-6), fine-tuning the last three layers (layers 4-6), and fine-tuning all of them (layers 1-6). Table 6 shows the results for each of these options.

Note that fine-tuning the last layer of DistilBERT results in an AUC score similar to those obtained with feature extraction, while all the other fine-tuning options yield significantly worse results. This is due to the learning rate being too high. The weights of the pre-trained language model are close to their optimal values, while those of the embeddings and layers 1,

Model	AUC	F_1	Time
NN+DistilBERT(fin. 6)	**0.8515**	**0.4819**	1:55:05
NN+BERT(fin. 6)	0.8500	0.4611	2:39:40
NN+DistilBERT(fin. 5-6)	0.8464	0.4548	1:41:14
NN+BERT(fin. 5-6)	0.8454	0.4534	2:38:22
NN+DistilBERT(fin. 4-6)	0.8461	0.4446	1:56:34
NN+DistilBERT(fin. 1-6)	0.8453	0.4330	2:48:41

Table 6: Fine-tuning results.

2, and 3 are randomly initialized and need a higher learning rate. A more robust fine-tuning alternative is presented in the next section. Note as well that fine-tuning the models takes longer than using feature extraction, and that the difference in time needed to fine-tune the last layers of BERT and DistilBERT (while keeping the rest of their layers frozen) is smaller than the one observed between them when performing feature extraction. Again, the times are given in *hh:mm:ss* format.

6.6. Stack-and-Finetune

The stack-and-finetune training strategy is as follows. First, we train the entire neural network (keeping the weights of the language model frozen) until convergence with the same learning rate as before, i.e., 3e-4. Then, the whole neural network is fine-tuned (including the language model) during a few epochs with a lower learning rate. This prevents catastrophic forgetting from happening, a phenomenon that might occur when language models are trained with a high learning rate. Table 7 shows our model results after using the stack-and-finetune approach with a learning rate of 1e-5 for the fine-tuning stage. The execution times of the stacking and the fine-tuning stages are reported separately in *hh:mm:ss* format.

Model	AUC	F_1	Time
NN+DistilBERT(lr=1e-5)	0.8518	**0.4847**	1:08:37 + 1:48:01
NN+BERT(lr=1e-5)	**0.8525**	0.4789	1:18:40 + 4:22:04

Table 7: Stack-and-finetune results.

The AUC and F_1 scores are higher (for both DistilBERT and BERT) than the scores obtained with feature extraction and fine-tuning alone. By using a higher learning rate at a first stage and a lower one thereafter, we were able to avoid catastrophic forgetting. Furthermore, the model with BERT achieves a higher AUC score than with DistilBERT, which is in line with BERT being a larger language model. However, note that stack-and-finetune the model with BERT takes twice as long as with DistilBERT. Last, we highlight that using the stack-and-finetune approach results in AUC scores of 0.8518 and 0.8525 (with DistilBERT and BERT, respectively), which is substantially higher than that of the model without a pre-trained language model (0.8459).

Considering the stack-and-finetune approach with BERT word embeddings, Table 8 compares our model with the SLAM-related techniques introduced in Section 2. Even though our model ranks in the seventh position, our main aim is to show that the pre-trained language models are useful for the SLAM task and prove that model distillation might enhance the efficiency much more than affect the effectiveness (one can note that the differences between AUC and F_1 are relatively small).

	Method	AUC	F_1
1	Hu et al. (2020)	0.864	0.564
2	Ruan et al. (2021)	0.863	0.564
3	Osika et al. (2018)	0.861	0.561
4	Xu et al. (2018)	0.861	0.559
5	Rich et al. (2018)	0.859	0.468
6	Sense et al. (2021)	0.854	-
7	Our method	0.853	0.479
8	Kaneko et al. (2018)	0.848	0.476
9	Bestgen (2018)	0.846	0.414
10	Yuan (2018)	0.841	0.479

Table 8: Comparition between our method and the methods proposed for the SLAM shared task.

7. Conclusion

We provided evidence that transfer learning through using pre-trained Language Models such as BERT and DistilBERT is effective for Second Language Acquisition Modeling. The

stack-and-finetune approach is preferred in terms of AUC and F_1 scores among the three methods presented. However, it must be noted that these improvements come at the expense of a larger computational cost and longer training times.

We explored three different ways of extracting and adapting the information stored in the pre-trained language model (DistilBERT or BERT). *Feature extraction* of the lowest two layers achieved a substantially higher AUC score and F_1-score than our base model (without a pre-trained language model); 0.8512 and 0.4772, respectively, for DistilBERT. Extracting the features of the last layers resulted in a worse performance, which is in line with the fact that upper layers learn more task-specific information. We argued that our corpus is much different from the corpora on which BERT was trained as our sentences are shorter and less syntactically complex. *Fine-tuning* the layers of the pre-trained language model did not show an improvement over feature extraction — fine-tuning the last layer of DistilBERT yielded an AUC score of 0.8515, while fine-tuning more layers damaged the model's performance due to the learning rate being too high. As a more robust fine-tuning alternative, we lastly showed the *stack-and-finetune* approach, which first freezes the pre-trained language model and then fine-tunes the whole model with a lower learning rate. This resulted in an AUC score of 0.8520 when using DistilBERT and a learning rate of 1e-6 for the fine-tuning stage, as well as an AUC score of 0.8525 when using BERT and a learning rate of 1e-5 for the fine-tuning stage.

Our work can be expanded in several directions. Regarding the data, we believe it would be interesting to make available and analyze the answers of more advanced language learners. It can be expected that, with more complex data (i.e., longer, more complex sentences), the benefits of using pre-trained language models such as BERT will be greater, as these models are trained on huge corpora with long, syntactically complex sentences. Regarding the model, a next step could be comparing the performance of pre-trained language models of smaller sizes in order to determine the trade-off between model capacity and performance for SLAM.

References

Andor, D., Alberti, C., Weiss, D., Severyn, A., Presta, A., Ganchev, K., Petrov, S., & Collins, M. (2016). Globally normalized transition-based neural networks. In 54th Annual

Meeting of the Association for Computational Linguistics (ACL 2016), 2442–2452. ACL. https://doi.org/10.18653/v1/P16-1231

Bestgen, Y. (2018). Predicting second language learner successes and mistakes by means of conjunctive features. In NAACL-HLT Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2018), 349–355. ACL. https://doi.org/10.18653/v1/w18-0542.

Bucilă, C., Caruana, R., & Niculescu-Mizil, A. (2006). Model compression. In 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2006), 535–541. ACM. https://doi.org/10.1145/1150402.1150464.

Cordova, D. I., & Lepper, M. R. (1996). Intrinsic motivation and the process of learning: Beneficial effects of contextualization, personalization, and choice. Journal of Educational Psychology, 88, 715–730. https://doi.org/10.1037/0022-0663.88.4.715.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2019), 1, 4171–4186. ACL. https://doi.org/10.18653/v1/N19-1423.

Ferreira-Mello, R., André, M., Pinheiro, A., Costa, E., & Romero, C. (2019). Text mining in education. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 9. https://doi.org/10.1002/widm.1332.

Gardner, R. C. (2014). Attitudes and motivation in second language learning. In Bilingualism, Multiculturalism, and Second Language Learning, 63–84. Psychology Press.

Goldberg, Y. (2019). Assessing BERT's syntactic abilities. arXiv preprint arXiv:1901.05287.

Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531.

Howard, J., & Ruder, S. (2018). Universal language model fine-tuning for text classification.

In 56th Annual Meeting of the Association for Computational Linguistics (ACL 2018), 328–339. ACL. https://doi.org/10.18653/v1/P18-1031.

Hu, Y., Huang, H., Lan, T., Wei, X., Nie, Y., Qi, J., Yang, L., & Mao, X. (2020). Multi-task Learning for Low-Resource Second Language Acquisition Modeling. In Asia-Pacific Web and Web-Age Information Management Joint International Conference on Web and Big Data (APWeb-WAIM 2020), 603-611. Springer. https://doi.org/10.1007/978-3-030-60259-8_44.

Kaneko, M., Kajiwara, T., & Komachi, M. (2018). TMU system for SLAM-2018. In NAACLHLT Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2018), 365–369. ACL. https://doi.org/10.18653/v1/w18-0544.

Li, L., Li, C., & Ji, D. (2021). Deep context modeling for multi-turn response selection in dialogue systems. Information Processing & Management, 58. https://doi.org/10.1016/j.ipm.2020.102415.

Meškelė, D., & Frasincar, F. (2020). Aldonar: A hybrid solution for sentence-level aspectbased sentiment analysis using a lexicalized domain ontology and a regularized neural attention model. Information Processing & Management, 57. https://doi.org/10.1016/j.ipm.2020.102211.

Mihaescu, M. C., & Popescu, P. S. (2021). Review on publicly available datasets for educational data mining. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 11. https://doi.org/10.1002/widm.1403.

Mohri, M. (1997). Finite-state transducers in language and speech processing. Computational Linguistics, 23 , 269–311. MIT Press.

Osika, A., Nilsson, S., Sydorchuk, A., Sahin, F., & Huss, A. (2018). Second language acquisition modeling: An ensemble approach. In NAACL-HLT Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2018), 217–222. ACL. https://doi.org/10.18653/v1/W18-0525.

Peters, M., Ruder, S., & Smith, N. A. (2019). To tune or not to tune? Adapting pretrained

representations to diverse tasks. In 4th Workshop on Representation Learning for NLP (RepL4NLP 2019), 7–14. ACL. https://doi.org/10.18653/v1/W19-4302.

Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training. OpenAI blog.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. OpenAI blog.

Rich, A., Popp, P. O., Halpern, D., Rothe, A., & Gureckis, T. (2018). Modeling secondlanguage learning from a psychological perspective. In NAACL-HLT Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2018), 223–230. ACL. https://doi.org/10.18653/v1/w18-0526.

Ruan, S., Wei, W., & Landay, J. (2021). Variational Deep Knowledge Tracing for Language Learning. In 11th International Learning Analytics and Knowledge Conference (LAK 2021), 323-332. ACM. https://doi.org/10.1145/3448139.3448170.

Ruder, S. (2019). Neural transfer learning for natural language processing. Ph.D. thesis NUI Galway.

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

Sense, F., Wood, R., Collins, M. G., Fiechter, J., Wood, A., Krusmark, M., Jastrzembski, T., & Myers, C. W. (2021). Cognition-Enhanced Machine Learning for Better Predictions with Limited Data. Topics in Cognitive Science. Wiley Online Library. https://doi.org/10.1111/tops.12574.

Settles, B. (2018). Data for the 2018 Duolingo Shared Task on Second Language Acquisition Modeling (SLAM). https://doi.org/10.7910/DVN/8SWHNO.

Settles, B., Brust, C., Gustafson, E., Hagiwara, M., & Madnani, N. (2018). Second language acquisition modeling. In NAACL-HLT Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2018), 56–65. ACL. https://doi.org/10.18653/v1/W18-0506.

Srivastava, M., & Goodman, N. (2021). Question generation for adaptive education. In 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, (ACL/IJCNLP 2021), 692–701. ACL. https://doi.org/10.18653/v1/2021.acl-short.88.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research, 15, 1929–1958.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. In 31st Annual Conference on Neural Information Processing Systems (NIPS 2017), 5998–6008. Curran Associates.

Wang, A., Pruksachatkun, Y., Nangia, N., Singh, A., Michael, J., Hill, F., Levy, O., & Bowman, S. R. (2019a). SuperGLUE: A stickier benchmark for general-purpose language understanding systems. In 32nd Annual Conference on Neural Information Processing Systems (NIPS 2019), 3261–3275.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., & Bowman, S. R. (2018). GLUE: A multi-task benchmark and analysis platform for natural language understanding. In Analyzing and Interpreting Neural Networks for NLP (EMNLP 2018) 353–355. ACL. https://doi.org/10.18653/v1/W18-5446.

Wang, R., Su, H., Wang, C., Ji, K., & Ding, J. (2019b). To tune or not to tune? How about the best of both worlds? arXiv preprint arXiv:1907.05338.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M. et al. (2019). Transformers: State-of-the-art natural language processing. arXiv preprint arXiv:1910.03771. https://doi.org/10.18653/v1/2020.emnlp-demos.6.

Wu, M., Davis, R. L., Domingue, B. W., Piech, C., & Goodman, N. D. (2020). Variational item response theory: Fast, accurate, and expressive. In 13th International Conference on Educational Data Mining (EDM 2020). IEDMS.

Xu, S., Chen, J., & Qin, L. (2018). CLUF: A neural model for second language acquisition modeling. In NAACL-HLT Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2018), 374–380. ACL. https://doi.org/10.18653/v1/W18-0546.

Young, T., Hazarika, D., Poria, S., & Cambria, E. (2018). Recent trends in deep learning based natural language processing. IEEE Computational Intelligence Magazine, 13 , 55–75. https://doi.org/10.1109/MCI.2018.2840738.

Yuan, Z. (2018). Neural sequence modelling for learner error prediction. In NAACL-HLT Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2018), 381–388. ACL. https://doi.org/10.18653/v1/W18-0547.

Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A., & Fidler, S. (2015). Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In IEEE International Conference on Computer Vision (ICCV 2015), 19–27. IEEE. https://doi.org/10.1109/ICCV.2015.11.