DistillEmb: Distilling Word Embeddings via Contrastive Learning

Amanuel Mersha
School of Information Technology and Engineering
Addis Ababa Institute of Technology
Addis Ababa, Ethiopia
amanuel.negash@aau.edu.et

Stephen WU
University of Texas Health Science Center at Houston
Houston, TX
Stephen.T.Wu@uth.tmc.edu

Abstract

Word embeddings powered the early days of neural network-based NLP research. Their effectiveness in small data regimes makes them still relevant in low-resource environments. However, they are limited in two critical ways: linearly increasing memory requirements and out-of-vocabulary token handling. In this work, we present a distillation technique of word embeddings into a CNN network using contrastive learning. This method allows embeddings to be regressed given the characters of a token. It is then used as a pretrained layer, replacing word embeddings. Low-resource languages are the primary beneficiary of this method and hence, we show its effectiveness on two morphology-rich Semitic languages, and in a multilingual NER task comprised of 10 African languages. Apart from being data and memory efficient, the model significantly increases performance across several benchmarks and is capable of transferring word representations.

1 Introduction

In the last decade, deep learning and the abundance of data have propelled innovation in NLP research. Word embedding algorithms such as Word2vec [19], Glove [22] and FastText [4] learn distributed word representation by exploiting co-occurrence frequency. While, word embeddings are shallow and only a word-level representation, deep contextualized models such as ELMo [23] learn richer word representation by predicting the next word from a sequence using deeper and larger models. Later, Transformer [32] based language models such as BERT[7] that employ the attention mechanism instead of recurrence showed superior performance on several benchmarks.

Unfortunately, this latest success that is based on large language models does not benefit low-resource languages, as these languages are often constrained by one or a combination of limited data and computing power. While the effort around training such models on several low-resource languages is promising [21, 30], they still are inaccessible for the target users as the cost of training them as well as deployment is very high. Overall, even though these models have many desirable properties, the cost of training and deploying them makes them impractical in lesser-resourced environments. Hence word embeddings continue to be critical in dealing with small compute and datasets. Despite their successes on small data regimes, however, word embeddings are not completely problem-free either. As each token requires its own vector representation, the total memory

Workshop on Transfer Learning for Natural Language Processing, 2022, New Orleans.

requirement increases linearly with the number of tokens. In addition, due to this one-to-one representation scheme, tokens that do not exist in the corpus are replaced by an *"unk" (unknown)* token.

Moreover, morphology-rich languages such as Amharic come with unique challenges when learning word representations. Due to many forms, words in such languages have lower frequency when compared to words in languages such as English. Unless the data is sufficiently large, algorithms that exploit the n-gram frequency will fail to capture the true relationship between words. For instance, in a parallel translation corpus, the English corpus has an average word frequency of 19 while the Amharic one has only 6. Moreover, the total amount of tokens in the English corpus is 13M with 67.27k unique tokens, while the Amharic one has less than 1M tokens with 145.2k unique tokens. The Amharic corpus is more compact than the English one as it contains a large number of unique tokens.

The heavy tail of these infrequent tokens becomes a major problem for word embedding algorithms as they rely on exploiting co-occurrence frequency. FastText mitigates this by utilizing the sub-word information to augment the main word representation [4]. However, it still suffers from the out-of-vocabulary problem in an online setting. Furthermore, the memory space requirement is now exacerbated as the total number of unique tokens is multiplied by several folds when compared to an English corpus. Similar problems also arise in multi-lingual settings. Large language models such as BERT solved this out-of-vocabulary problem by jointly learning subword embeddings [27, 13]. These subword embeddings are also problematic when it comes to memory usage under a low resource setting. For morphologically complex languages and multi-lingual settings, the unique subword vocabulary easily reaches 30k. Furthermore, there is no clear way of learning the subword embeddings without deeper models.

Hence, we asked *Could there be a model that compromises between the two approaches?* Specifically, could a model be both memory and compute-efficient, while providing the flexibility of contextualized models to avoid the out-of-vocabulary problem?

In this work, we introduce *DistillEmb*, a CNN network that distills word embedding knowledge. This scheme compresses the entire word embedding set into a four-layer CNN that regresses a word embedding given the word's characters. Then this distilled, pretrained model is used as an embedding layer in downstream tasks. The benefits are (a) no out-of-vocabulary problem, (b) a drastically smaller, constant memory cost, (c) better performance compared to word embeddings, and (d) out-of-the-box cross-lingual word representation transferability. One should note that since indexing an embedding is replaced by computing the embedding, the resulting model is slower than word embeddings, though it is still faster than large models. Through extensive experiments, we show that this model sits between the shallow and large deep models.

2 Related Work

Since large models require significant memory and computing power, model compression has been an active area of study. Pruning, quantization, low-rank factorization, and other methods were introduced to compress models. In this work, we focus on the knowledge distillation technique [10] in which a student network learns usually from a more powerful teacher network.

Sanh et al. [25] showed that large models such as BERT [7] can be distilled into a single-layer BiLSTM model with a small performance drop. Tang et al. [29] also showed that BERT can be distilled into its smaller BERT version that weighs only 40% of the original size but is 60% faster.

In the space of word embeddings, to our knowledge, distilling them has not been studied. Instead, compression techniques through other methods were proposed. Raunak et al. [24] explored dimensionality reduction of word embeddings using PCA. In this method, first, a simple post-processing step of removal of common vectors is applied similar to Mu et al. [20]. Then, PCA is applied to reduce the dimensionality followed by the post-processing introduced by Mu et al. [20]. The technique enabled the compression of a 300 vector into 150. Regarding accuracy on downstream tasks, the compressed embedding performed similarly or better when compared to the original Glove and FastText embeddings. Andrews [2] used Lloyd's algorithm to compress word embeddings through quantization. They showed that Glove embeddings [22] can be compressed by a factor of 10% with negligible loss of performance.

Shu and Nakayama [28] proposed a learning code that represents a word using a neural network instead of the actual word. Each word has a compact code that is composed of integers in which similar words have similar codes. Using the compact code, learned embeddings of the components of the codes are summed up to give a word representation. The number of learned embeddings is much smaller than the original number of embeddings. Lee et al. [14] applied the *GroupReduce* concept that was applied on language models [5] on word embeddings with better weighting and clustering of tokens.

These methods, however, still suffer from out-of-vocabulary issues and linear memory requirements. In this work, we pretrain a small constant-dimensionality CNN model through word embedding knowledge distillation. Once this model is trained, it replaces word embeddings in downstream tasks. CNN has been shown to improve word representation by learning embeddings directly from characters [23, 11, 12, 15] while lowering the memory cost as the number of the learned embeddings is as large as the number of characters.

3 Method

Our proposed method *DistillEmb* is to distill learned word embeddings into a convolutional neural network. The ability of a neural network to approximate any function motivated the idea that given the characters of a word, a neural network should be able to regress the word's meaning in a high-dimensional space. The overall process is: (1) train word embedding using a corpus, (2) distill the knowledge into a CNN, (3) use the trained CNN as a pretrained word representation layer in the downstream task.

3.1 Corpus and Raw embeddings

We will use *Raw embedding* to refer to the traditional embeddings such as word2vec and FastText. In the Amharic language case, the corpus is collected from various web sources such as news websites. After preprocessing, the corpus contains 4.6M unique tokens with only 19% of them having a frequency of more than 4. 5 is the minimum token frequency set in the word2vec and FastText training settings. In total, the corpus contains 153M tokens. The preprocessing step is the following:

- Replace characters other than the valid character set by "-". Our valid characters are Amharic digits, Amharic alphabet, Amharic punctuation, English digit, and English punctuation.
- Replace repetitive tokens that are only made of "-" with a single "<—>"

The Tigrinya languages corpus [31] contains 35.5M tokens with 1.6M of them unique. The above pre-processing step is used to clean the dataset. We then trained a word2vec [19] and FastText [4] models using the corpses. Apart from being distilled into the CNN network, these embeddings are also used as a baseline for our evaluations.

For the multilingual case, we directly employed the pretraining corpus from Ogueji et al. [21]. After preprocessing, the corpus contains 5.6M unique tokens with only 15% of them having a frequency of more than 4. In total, it contains 140M tokens. The preprocessing step is implemented in the following way:

- Compute the frequency of each character
- Select the characters with a frequency of more than 99 as valid characters
- Replace invalid characters with "-"

After applying the above operation, out of 5391 characters, only 739 (13.9%) remained in the dataset. A large majority of the characters don't contribute much to the corpus. We trained word2vec and FastText embeddings to train the multilingual CNN network.

3.2 Student model - CNN

The CNN model is constructed using a character embedding layer followed by a set of four one-dimensional convolution-pooling layers. The character embedding layer has a size of 64. We set the maximum character length of a word to 13 after observing the length of the words. Hence, an

input to the CNN layer has the shape of 64×13. A 1d convolution is applied across the embedding dimension. Finally, a fully connected layer maps the output of the last convolutional layer to the traditional 300-length vector output. This output is the vector that is used for both the distillation as well as the downstream tasks.

Given the fact that LSTMs learn sequence mapping, it might seem that they would be better suited for this task. In our preliminary evaluation, however, the distillation error rate barely decreases for the LSTM-based model despite having two bidirectional layers. Varying the hidden size also did not help improve the error. Hence, CNN was chosen as it is easier to train and faster due to its parallelizability.

3.3 Triplet Loss

Our main goal is to train the CNN so that it outputs the embedding of the word given its characters. A simple way to achieve this is to train the model to generate embeddings that minimize the mean squared error against the raw embeddings. A preliminary experiment showed that this mechanism results in very poor performance.

Instead of directly minimizing the mean squared error we can train the model to capture the similarities and differences between words. Contrastive learning is a mechanism by which a model learns representations by comparing similar and opposite inputs. It has been successful in face detection. Triplet loss is a type of contrastive loss where the model tries to minimize the Euclidean distance between two similar data points and maximize their distance for unrelated data points.

In DistillEmb's case, triplet loss is applied in the following way: given a word, the model predicts an embedding that will be the anchor. The positive example is the raw embedding of that word. The negative example can be any of the embeddings other than the positive one. However, it is essential to carefully select a negative example so that the model learns meaningful differences between the inputs. An ideal negative example is one that is very similar to the positive embedding. This is because such combinations force the model to learn even the most minute differences between the pair. A typical way to extract such a set of negative examples for each token is to compare a positive embedding to all other embeddings, which is a very expensive operation. Hence we employed a simple approximation algorithm described below:

- From the corpus, randomly select a sequence of tokens with a length of N. We found $N = 32$ to be enough.
- Extract the raw embedding for each token in the sequence
- Compute their cosine similarity against the target positive embedding
- Pick the embedding with the highest cosine similarity

Some words are similar in meaning but are written with wildly different characters. Some have similar characters but are different in meaning. Fortunately, evaluation on the Amharic word analogy task [18] showed that word2vec retains high semantic information and FastText retains character-level information. Hence, the cosine similarity is computed with the concatenated vector of the two. This allows the algorithm to find better negative examples when compared to using just one type of embedding. Below is the pseudo-code of the triplet loss implementation.

w = CNN(characters)
$w_p = concat(word2vec[w_i], FastText[w_i])$
$w_n = negative\_mine(W, w_p)$
$L(w, w_p, w_n) = max(|w - w_p|^2 - |w - w_n|^2 + \alpha, 0)$

W is the set containing all the embedding of the tokens. w is the anchor which is also an embedding for a word. $L(w, w_p, w_n)$ is the triplet loss with margin $\alpha = 1$.

3.4 Training the model

In the distillation phase, we found a slight edge in using both FastText and Word2vec embeddings to compute the triplet loss. Two triplet losses - one based on word2vec and another based on the FastText embeddings - are computed on the same anchor. Then the two losses are averaged and

Model	Sentiment	POS	NER	News
Word2vec-BiLSTM (ours)	47.75	82.42	41.78	78.60
FastText-BiLSTM (ours)	51.50	82.43	41.47	81.00
DistillEmb (ours)	54.13	89.78	57.13	**82.16**
Flair-word2vec	55.22	81.05	62.34	-
Flair-FastText	53.00	83.94	65.24	-
Flair-Contextual	**56.92**	**94.08**	**77.61**	-

Table 1: Evaluation of BiLSTM-DistillEMB test F1 score. Word2vec-BiLSTM and FastText-BiLSTM are trained with the same raw embedding from which DistillEmb is distilled from. The Flair-based models are evaluated by Yimam et al. [33]. There is no evaluation of the news text classification as it is new. Best values are **bold** and our DistillEmb evaluations are underlined.

back-propagated using an AdamW optimizer [17]. All of the models were trained with an initial learning rate of 0.001 and a step scheduler of $gamma = 0.95$. The batch size is set to 64.

4 Evaluation

After the distillation is done, the resulting model simply replaces the word embedding layer and is fine-tuned in the downstream task. Hence, the output of the last layer is fed to the target task network which is usually a multi-layer BiLSTM. In our downstream tasks, except for the multi-lingual NER task, single-layer BiLSTM networks were employed. Each of the experiments is run 5 times without seeding the random generator. The validation dataset was used to select a model. At each epoch, the F1 score of each experiment is collected. We then averaged the F1-Score across the 5 trials. The epoch that had the maximum average F1 score was picked. Then we evaluate the 5 models on the test set at that epoch. The final F1-Score is the average of those 5 test F1 scores. All of the models are tested in such a way and the test values we present below are not the best model, but the average from 5 models. We hope that this will reduce the reproducibility problem.

4.1 Amharic Benchmark

Sentiment Analysis: The dataset for the sentiment is published by Yimam et al. [33]. However, they only published the Twitter IDs of the tweets and we had to scrap them again. Sadly, 20.89% of the data was already lost from the Twitter website. Thus, we merged all the train, test, and validation sets and re-sampled them into a 70-10-20 ratio for training, validating, and testing purposes. The model used for this task is a single-layer bidirectional LSTM that has a 256 hidden size. The complete parameters of the experiment are stated in Appendix A.

Part of Speech Tagging: The original POS data was published by Ethiopian Language Research Center (ELRC), Addis Ababa University Demeke and Getachew [6] and revised by Gashaw and Shashirekha [9]. In the revision, the authors included annotated texts from Quranic and Biblical texts referred to as ELRCQB. It contains 39k sentences with 62 tags. In this experiment, the split 80-10-10 used by Yimam et al. [33] is employed.

Named Entity Recognition: The NER dataset is published originally by New Mexico State University on GitHub. The dataset contains 4237 sentences where 5480 tokens are tagged out of 109k. It was used by Ogueji et al. [21] with the split of 70-10-20 ratio. In this experiment, their version of the dataset is used for the experiment.

News Text Classification: We used the dataset published by Azime and Mohammed [3] and re-sampled it to make it comparable with the BBC news classification dataset. It has 5 classes and a total of 2225 news samples, split into 70-10-20 ratio for training, validating, and testing. We used the same model specified in the sentiment classification tasks.

Table 1 shows the performance of Raw Embedding + BiLSTM, DistillEmb + BiLSTM, and Flair [26] based models that were trained on Amharic Corpus by Yimam et al. [33]. The Flair-Contextual model uses contextual embedding collected from the Amharic pretrained RoBERTa model [16].

Language	CNN-BiLSTM CRF	AfriBERTa Small (97M)	AfriBERTa Base (111M)	AfriBERTa Large (126M)	DistillEmb-BiLSTM-CRF (3.3M) (ours)
amh	52.89	67.90	71.80	**73.82**	56.31
hau	83.70	89.01	90.10	**90.17**	86.50
ibo	78.48	86.63	86.70	**87.38**	83.45
kin	64.61	69.91	73.22	**73.78**	70.71
lug	74.31	76.44	**79.30**	78.85	79.19
luo	66.42	67.31	70.63	70.23	**73.56**
pcm	66.43	82.92	84.87	**85.70**	79.43
swa	79.26	85.68	**88.00**	87.96	82.50
wol	60.43	60.10	61.82	61.81	**64.75**
yor	67.07	76.08	79.36	**81.32**	77.59
Ave.	69.36	76.20	78.60	**79.10**	77.58

Table 2: DistillEmb test F1 score compared to previous work. The four other models are evaluated by Ogueji et al. [21]. Despite being a very small model, DistillEmb-BiLSTM-CRF has outperformed AfriBERTa-Small in 5 languages with a higher average score. It also achieves the highest accuracy.

Model	33%	66%	100%
DistillEmb	78.09	77.89	77.58

Table 3: Multilingual DistillEmb averaged test F1 score over the 10 languages. The full performance report is listed in Table 8, Appendix section

Overall, DistillEmb-BiLSTM outperforms all other raw embedding-based models. Compared to the contextual-based models, DistillEmb falls behind on Sentiment Analysis and NER tasks. Note that, in the Sentiment Analysis task, it is trained on 14% lesser data size.

4.2 Multilingual embedding

To determine if the method is applicable in a multilingual scenario, we evaluated it on a multilingual NER dataset that contains 10 African languages. We used the 1GB multilingual corpus introduced by Ogueji et al. [21]. The authors evaluated their Transformer based model called AfriBERTa on 10 NER tasks of different languages assembled by Adelani et al. [1]. As a baseline for this evaluation, we compare our results to Ogueji et al. [21], Table 7.

Table 2 shows that the F1 score of CNN-BiLSTM-CRF, the AfriBERTa [21] and DistillEmb-BiLSTM-CRF models. Technically, the CNN-BiLSTM-CRF model and DistillEmb-BiLSTM-CRF are similar. However, the pretraining of DistillEmb made a large difference. It added almost 8% aggregated F1 score. Furthermore, it surpassed the AfriBERTa-small model which has 97M parameters by almost 3%.

We further evaluated how the model performs when trained on different pretraining data sizes, specifically on 33%, 66%, and 100%. The result is very surprising as increasing the data size actually hurts the performance. Table 3 shows the overall performance of the model on the data sizes. The full list of evaluations can be found in Table 8. The model performed best on 33% data size, which is very close to the performance of the AfriBERTa-base model. This shows that DistillEmb is highly data efficient.

4.3 Cross-lingual Transferability

As the model learns representations at a character-level which are then aggregated into a word-level, it presents a unique opportunity of transferring knowledge between similar languages under a low-resource setting. Such a feature becomes handy when one has a larger corpus in one language but little to no data for a similar language. To test this feature, we chose the Tigrinya language. Apart from being a Semitic language, it has many commonalities with Amharic as both are descendants

Model	Data-33%	Data-66%	Data-100%
TigXLNet [31]	-	-	83.29
Random	81.97	85.12	85.96
Am-Distill	**84.26**	86.32	87.50
Tig-Distill	84.22	86.25	**87.91**
FastText	80.25	**87.41**	**87.91**

Table 4: Cross-lingual transfer test on Tigrinya Sentiment Analysis.

of Ge'ez. Feleke [8] showed that Tigriyna has a phonetic distance of about 30% when compared to Amharic.

We used the Sentiment Analysis task introduced by Tela et al. [31]. The original dataset contained about 50k samples for training and 4k for testing. We re-sampled it into a 70-10-20 ratio for training, validation, and testing. To quantify how much information from the Amharic language transferred to Tigrinya through DistillEmb, we trained the sentiment model in four scenarios: (a) using FastText embeddings (FastText), (b) DistillEmb initialized randomly (Random), (c) DistillEmb trained on an Amharic corpus (Am-DistillEmb) and (d) DistillEmb trained on a Tigrinya corpus (Tig-DistillEmb). Furthermore, we tested all of the models on 33%, 66%, and 100% task train data size to determine the relationship between the pretraining and the downstream task data size.

Table 4 shows the test F1 scores of the models on different data sizes. Under a low data regime (33%), it's evident that even the randomly initialized DistillEmb model (Random) performs better than the FastText embedding-based classifier. The Am-DistillEmb model shows the highest performance on the same data size. This is because it's trained on the larger Amharic corpus and the knowledge is being utilized in the downstream task. Even in the full data size case, its performance closes on the Tig-DistillEmb and Tigrinya FastText models.

4.4 Cost of DistillEmb

The number of parameters of DistillEmb is 950k. Comparing DistillEmb-based models with the word embedding counterparts, they take 14.3$\times$ lower memory space, a 1330% decrease.

A typical word embedding-based model uses an indexing operation while processing a word meaning, and thus has no computation cost. The DistillEmb model, however, relies on computation to produce the word meaning and, thus is costly. Benchmarking both kinds of models in different batch sizes, overall, DistillEmb is 4.25$\times$ slower when compared to a model that uses word embeddings. This throughput test is done on GTX-940MX which is a very low-end laptop GPU. To mitigate this problem, caching can be implemented to save the compute time for the most used tokens.

5 Limitations

One has to first train a word embedding model and then distill it later. This is basically pretraining for pretraining and might be an extra cost. Furthermore, not only the model takes a longer time for each epoch in the training phase, but it also needs a higher number of epochs generally. Performance and memory usage have been greatly improved but the throughput has declined when compared to embedding-based models. It is highly advisable to do a cost-benefit analysis of the technique before deployment.

6 Conclusion

In this paper, we proposed a CNN-based model called DistillEmb that is pretrained through the distillation of raw embeddings. It works by regressing the embedding of a word by processing the word's characters. It primarily handles out-of-vocabulary issues and linear memory requirements. Extensive experiments show that on four Amharic tasks, the model improves performance. This is due to its capacity to interpolate the representation of unseen words. The evaluation of the 10 multilingual NER shows that the model is data efficient and performs well in multilingual settings. Finally, we showed that DistillEmb enables cross-lingual word representation transfer out of the box through extensive experiments on the Amharic-Tigrinya language pair. While it significantly

decreases memory requirement, a slight throughput decline is observed in all tasks. We hope that low-resource languages benefit from such techniques as the variety of experiments show that the technique is well suited for these languages.

References

[1] David Ifeoluwa Adelani, Jade Z. Abbott, Graham Neubig, Daniel D'souza, Julia Kreutzer, Constantine Lignos, Chester Palen-Michel, Happy Buzaaba, Shruti Rijhwani, Sebastian Ruder, Stephen Mayhew, Israel Abebe Azime, Shamsuddeen Hassan Muhammad, Chris C. Emezue, Joyce Nakatumba-Nabende, Perez Ogayo, Anuoluwapo Aremu, Catherine Gitau, Derguene Mbaye, Jesujoba Oluwadara Alabi, Seid Muhie Yimam, Tajuddeen R. Gwadabe, Ignatius U. Ezeani, Rubungo Andre Niyongabo, Jonathan Mukiibi, Verrah A Otiende, Iroro Orife, Davis David, Samba Ngom, Tosin P. Adewumi, Paul Rayson, Mofetoluwa Adeyemi, Gerald Muriuki, Emmanuel Anebi, Chiamaka Ijeoma Chukwuneke, Nkiruka Bridget Odu, Eric Peter Wairagala, S. Ajiboye Oyerinde, Clemencia Siro, Tobius Saul Bateesa, Temilola Oloyede, Yvonne Wambui, Victor Akinode, Deborah Nabagereka, Maurice Katusiime, Ayodele Awokoya, Mouhamadane Mboup, Dibora Gebreyohannes, Henok Tilaye, Kelechi Nwaike, Degaga Wolde, Abdoulaye N Faye, Blessing K. Sibanda, Orevaoghene Ahia, Bonaventure F. P. Dossou, Kelechi Ogueji, Thierno Ibrahima Diop, Abdoulaye Diallo, Adewale Akinfaderin, Tendai Munyaradzi Marengereke, and Salomey Osei. Masakhaner: Named entity recognition for african languages. *Transactions of the Association for Computational Linguistics*, 9:1116–1131, 2021.

[2] Martin Andrews. Compressing word embeddings. In *ICONIP*, 2016.

[3] Israel Abebe Azime and Nebil Mohammed. An amharic news text classification dataset. *ArXiv*, abs/2103.05639, 2021.

[4] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[5] Patrick H. Chen, Si Si, Yang Li, Ciprian Chelba, and Cho-Jui Hsieh. Groupreduce: Block-wise low-rank approximation for neural language model shrinking. In *NeurIPS*, 2018.

[6] Girma A Demeke and Mesfin Getachew. Manual annotation of amharic news items with part-of-speech tags and its challenges. *Ethiopian Languages Research Center Working Papers*, 2 (1):1–16, 2006.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *ArXiv*, abs/1810.04805, 2019.

[8] Tekabe Legesse Feleke. The similarity and mutual intelligibility between Amharic and Tigrigna varieties. In *Proceedings of the Fourth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*, pages 47–54, Valencia, Spain, April 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-1206. URL `https://aclanthology.org/W17-1206`.

[9] Ibrahim Gashaw and H. L. Shashirekha. Machine learning approaches for amharic parts-of-speech tagging. *ArXiv*, abs/2001.03324, 2020.

[10] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *ArXiv*, abs/1503.02531, 2015.

[11] Rafal Józefowicz, Oriol Vinyals, Mike Schuster, Noam M. Shazeer, and Yonghui Wu. Exploring the limits of language modeling. *ArXiv*, abs/1602.02410, 2016.

[12] Yoon Kim, Yacine Jernite, David A. Sontag, and Alexander M. Rush. Character-aware neural language models. In *AAAI*, 2016.

[13] Taku Kudo and John Richardson. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *EMNLP*, 2018.

[14] Jong-Ryul Lee, Yong-Ju Lee, and Yong-Hyuk Moon. Block-wise word embedding compression revisited: Better weighting and structuring. In *EMNLP*, 2021.

[15] Wang Ling, Chris Dyer, Alan W. Black, Isabel Trancoso, Ramón Fernández Astudillo, Silvio Amir, Luís Marujo, and Tiago Luís. Finding function in form: Compositional character models for open vocabulary word representation. In *EMNLP*, 2015.

[16] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692, 2019.

[17] Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. *ArXiv*, abs/1711.05101, 2017.

[18] Amanuel Mersha and Stephen Wu. Morphology-rich alphasyllabary embeddings. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 2590–2595, Marseille, France, May 2020. European Language Resources Association. ISBN 979-10-95546-34-4. URL https://aclanthology.org/2020.lrec-1.315.

[19] Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *ICLR*, 2013.

[20] Jiaqi Mu, S. Bhat, and Pramod Viswanath. All-but-the-top: Simple and effective postprocessing for word representations. *ArXiv*, abs/1702.01417, 2018.

[21] Kelechi Ogueji, Yuxin Zhu, and Jimmy Lin. Small data? no problem! exploring the viability of pretrained multilingual language models for low-resourced languages. In *Proceedings of the 1st Workshop on Multilingual Representation Learning*, pages 116–126, Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.mrl-1.11. URL https://aclanthology.org/2021.mrl-1.11.

[22] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1162. URL https://aclanthology.org/D14-1162.

[23] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *NAACL*, 2018.

[24] Vikas Raunak, Vivek Gupta, and Florian Metze. Effective dimensionality reduction for word embeddings. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 235–243, Florence, Italy, August 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-4328. URL https://aclanthology.org/W19-4328.

[25] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108, 2019.

[26] Stefan Schweter and A. Akbik. Flert: Document-level features for named entity recognition. *ArXiv*, abs/2011.06993, 2020.

[27] Rico Sennrich, Barry Haddow, and Alexandra Birch. 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, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1162. URL https://aclanthology.org/P16-1162.

[28] Raphael Shu and Hideki Nakayama. Compressing word embeddings via deep compositional code learning. *ArXiv*, abs/1711.01068, 2018.

[29] Raphael Tang, Yao Lu, Linqing Liu, Lili Mou, Olga Vechtomova, and Jimmy J. Lin. Distilling task-specific knowledge from bert into simple neural networks. *ArXiv*, abs/1903.12136, 2019.

[30] Nllb team, Marta Ruiz Costa-jussà, James Cross, Onur cCelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, Anna Sun, Skyler Wang, Guillaume Wenzek, Alison Youngblood, Bapi Akula, Loïc Barrault, Gabriel Mejia Gonzalez, Prangthip Hansanti, John Hoffman, Semarley Jarrett, Kaushik Ram Sadagopan, Dirk Rowe, Shannon L. Spruit, C. Tran, Pierre Andrews, Necip Fazil Ayan, Shruti Bhosale, Sergey Edunov, Angela Fan, Cynthia Gao, Vedanuj Goswami, Francisco Guzm'an, Philipp Koehn, Alexandre Mourachko, Christophe Ropers, Safiyyah Saleem, Holger Schwenk, and Jeff Wang. No language left behind: Scaling human-centered machine translation. *ArXiv*, abs/2207.04672, 2022.

[31] Abrhalei Tela, Abraham Woubie, and Ville Hautamäki. Transferring monolingual model to low-resource language: The case of tigrinya. *ArXiv*, abs/2006.07698, 2020.

[32] Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *ArXiv*, abs/1706.03762, 2017.

[33] Seid Muhie Yimam, Abinew Ali Ayele, Gopalakrishnan Venkatesh, and Christian Biemann. Introducing various semantic models for amharic: Experimentation and evaluation with multiple tasks and datasets. *Future Internet*, 13:275, 2021.

A Models and Training Hyperparameters

B Distillation

Hyperparam	Values
Network	CNN
Layers	4
Hidden Size	300
Dropout	0.0
Batch Size	64
Seq-len (N)	32
Epochs	64
Optimizer	AdamW
LR Scheduler	StepLR
Scheduler gamma	0.95
Scheduler Step	1
Learning rate	0.001

Table 5: Hyper-paramaters for training the CNN through distillation. Increasing the batch size as well as sequence size make convergence very tough. Lowering them will result in poor performing model even if it has lower distillation error.

B.1 Raw Embedding Amharic NER, POS, Sentiment News Classification

Hyperparam	Values	Best
Network	LSTM	LSTM
Layers	1, 2	2
Hidden Size	256, 512	512
Dropout	0.4, 0.6	0.6
Batch Size	32, 64	64
Max Seq Len	200	200
Epochs	60	60
Optimizer	AdamW	AdamW
LR Scheduler	StepLR	StepLR
Scheduler gamma	0.96	0.96
Scheduler Step	1	1
Learning rate	0.001	0.01

Table 6: Hyper-params for embedding based training

B.2 DistillEmb based Amharic NER, POS, Sentiment News Classification, Multi-lingual

B.3 Multilingual NER Evaluation

Hyperparam	Values	Best
Network	LSTM	LSTM
Layers	1, 2	1
Hidden Size	256, 512	256
LSTM Dropout	0.1, 0.2	0.2
DistillEmb (CNN) Dropout	0.05	0.05
Batch Size	32, 64	64
Max Seq Len	200	200
Epochs	60	60
Optimizer	AdamW	AdamW
LR Scheduler	StepLR	StepLR
Scheduler gamma	0.98	0.98
Scheduler Step	1	1
Learning rate	0.0001	0.0001

Table 7: Hyper-params for DistillEmb based training. When using DistillEmb in downstream tasks, the model becomes very senstive to the dropout in the CNNs. Initially, in distillation, it should be trained with dropout = 0.0. But in downstream task, it should 0.05. Otherwise, increasing it will hurt performance and decreasing it will result in overfitting model.

Language	DistillEmb-BILSTM CRF (Data-33%)	DistillEmb-BILSTM CRF (Data-66%)	DistillEmb-BILSTM CRF (Data-100%)	AfriBERTa Large (126M)
amh	**56.91**	56.18	56.31	73.82
hau	86.68	**87.14**	86.50	90.17
ibo	83.95	**84.30**	83.45	87.38
kin	**71.23**	69.83	70.71	73.78
lug	80.13	**80.63**	79.19	78.85
luo	**73.80**	73.46	73.56	70.23
pcm	**80.46**	79.70	79.43	85.70
swa	**82.93**	82.85	82.50	87.96
wol	**65.78**	63.94	64.75	61.81
yor	77.14	**77.73**	77.59	81.32
Ave.	**78.09**	77.89	77.58	79.10

Table 8: Test F1 score of multi-lingual DistillEmb performance after trained on 33%, 66% and 100% of the pretraining corpus.