Language Modeling for Morphologically Rich Languages: Character-Aware Modeling for Word-Level Prediction

Daniela Gerz[1], Ivan Vulić[1], Edoardo Ponti[1]
Jason Naradowsky[3], Roi Reichart[2] Anna Korhonen[1]

[1]Language Technology Lab, DTAL, University of Cambridge
[2]Faculty of Industrial Engineering and Management, Technion, IIT
[3]Johns Hopkins University
[1]{dsg40,iv250,ep490,alk23}@cam.ac.uk
[2]roiri@ie.technion.ac.il
[3]narad@jhu.edu

Abstract

Neural architectures are prominent in the construction of language models (LMs). However, word-level prediction is typically agnostic of subword-level information (characters and character sequences) and operates over a closed vocabulary, consisting of a limited word set. Indeed, while subword-aware models boost performance across a variety of NLP tasks, previous work did not evaluate the ability of these models to assist next-word prediction in language modeling tasks. Such subword-level informed models should be particularly effective for morphologically-rich languages (MRLs) that exhibit high type-to-token ratios. In this work, we present a large-scale LM study on 50 typologically diverse languages covering a wide variety of morphological systems, and offer new LM benchmarks to the community, while considering subword-level information. The main technical contribution of our work is a novel method for injecting subword-level information into semantic word vectors, integrated into the neural language modeling training, to facilitate word-level prediction. We conduct experiments in the LM setting where the number of infrequent words is large, and demonstrate strong perplexity gains across our 50 languages, especially for morphologically-rich languages. Our code and data sets are publicly available.

1 Introduction

Language Modeling (LM) is a key NLP task, serving as an important component for applications that require some form of text generation, such as machine translation (Vaswani et al., 2013), speech recognition (Mikolov et al., 2010), dialogue generation (Serban et al., 2016), or summarisation (Filippova et al., 2015).

A traditional recurrent neural network (RNN) LM setup operates on a limited closed vocabulary of words (Bengio et al., 2003; Mikolov et al., 2010). The limitation arises due to the model learning parameters exclusive to single words. A standard training procedure for neural LMs *gradually* modifies the parameters based on contextual/distributional information: each occurrence of a word token in training data contributes to the estimate of a word vector (i.e., model parameters) assigned to this word type. Low-frequency words therefore often have incorrect estimates, not having moved far from their random initialisation. A common strategy for dealing with this issue is to simply exclude the low-quality parameters from the model (i.e., to replace them with the <*unk*> placeholder), leading to only a subset of the vocabulary being represented by the model.

This limited vocabulary assumption enables the model to bypass the problem of unreliable word estimates for low-frequency and unseen words, but it does not resolve it. The assumption is far from ideal, partly due to the Zipfian nature of each language (Zipf, 1949), and its limitation is even more pronounced for morphologically-rich languages (MRLs): these languages inherently generate a plethora of words by their morphological systems. As a consequence, there will be a large number of words for which a standard RNN LM cannot guarantee a reliable word estimate.

Since gradual parameter estimation based on contextual information is not feasible for rare phenomena

Transactions of the Association for Computational Linguistics, vol. 6, pp. 451–465, 2018. Action Editor: Brian Roark.
Submission batch: 12/2017; Revision batch: 5/2018; Published 7/2018.

in the *full vocabulary setup* (Adams et al., 2017), it is of crucial importance to construct and enable techniques that can obtain these parameters in alternative ways. One solution is to draw information from additional sources, such as characters and character sequences. As a consequence, such character-aware models should facilitate LM word-level prediction in a real-life LM setup which deals with a large amount of low-frequency or unseen words.

Efforts into this direction have yielded exciting results, primarily on the *input* side of neural LMs. A standard RNN LM architecture relies on two word representation matrices learned during training for its *input* and *next-word* prediction. This effectively means that there are two sets of per-word specific parameters that need to be trained. Recent work shows that it is possible to generate a word representation on-the-fly based on its constituent characters, thereby effectively solving the problem for the parameter set on the *input* side of the model (Kim et al., 2016; Luong and Manning, 2016; Miyamoto and Cho, 2016; Ling et al., 2015). However, it is not straightforward how to advance these ideas to the *output* side of the model, as this second set of word-specific parameters is directly responsible for the next-word prediction: it has to encode a much wider range of information, such as topical and semantic knowledge about words, which cannot be easily obtained from its characters alone (Jozefowicz et al., 2016).

While one solution is to directly output characters instead of words (Graves, 2013; Miyamoto and Cho, 2016), a recent work from Jozefowicz et al. (2016) suggests that such purely character-based architectures, which do not reserve parameters for information specific to single words, cannot attain state-of-the-art LM performance on word-level prediction.

In this work, we *combine* the two worlds and propose a novel LM approach which relies on both word-level (i.e., contextual) and subword-level knowledge. In addition to training word-specific parameters for word-level prediction using a regular LM objective, our method encourages the parameters to also reflect subword-level patterns by injecting knowledge about morphology. This information is extracted in an unsupervised manner based on already available information in convolutional filters from earlier network layers. The proposed method leads to large improvements in perplexity across a wide spectrum of languages: 22 in English, 144 in Hebrew, 378 in Finnish, 957 in Korean on our LM benchmarks. We also show that the gains extend to another multilingual LM evaluation set, compiled recently for 7 languages by Kawakami et al. (2017).

We conduct a systematic LM study on 50 typologically diverse languages, sampled to represent a variety of morphological systems. We discuss the implications of typological diversity on the LM task, both theoretically in Section 2, and empirically in Section 7; we find a clear correspondence between performance of state-of-the art LMs and structural linguistic properties. Further, the consistent perplexity gains across the large sample of languages suggest wide applicability of our novel method.

Finally, this article can also be read as a comprehensive multilingual analysis of current LM architectures on a set of languages which is much larger than the ones used in recent LM work (Botha and Blunsom, 2014; Vania and Lopez, 2017; Kawakami et al., 2017). We hope that this article with its new datasets, methodology and models, all available online at `http://people.ds.cam.ac.uk/dsg40/lmmrl.html`, will pave the way for true multilingual research in language modeling.

2 LM Data and Typological Diversity

A language model defines a probability distribution over sequences of tokens, and is typically trained to maximise the likelihood of token input sequences. Formally, the LM objective is expressed as follows:

$$P(t_1, ...t_n) = \prod_i P(t_i | t_1, ...t_{i-1}). \qquad (1)$$

t_i is a token with the index i in the sequence. For *word-level prediction* a token corresponds to one word, whereas for *character-level (*also termed *char-level) prediction* it is one character.

LMs are most commonly tested on Western European languages. Standard LM benchmarks in English include the Penn Treebank (PTB) (Marcus et al., 1993), the 1 Billion Word Benchmark (BWB) (Chelba et al., 2014), and the Hutter Prize data (Hutter, 2012). English datasets extracted from BBC News (Greene and Cunningham, 2006) and IMDB Movie Reviews (Maas et al., 2011) are also used for LM evaluation (Wang and Cho, 2016; Miyamoto and

Cho, 2016; Press and Wolf, 2017).

Regarding multilingual LM evaluation, Botha and Blunsom (2014) extract datasets for other languages from the sets provided by the 2013 Workshop on Statistical Machine Translation (WMT) (Bojar et al., 2013): they experiment with Czech, French, Spanish, German and Russian. A recent work of Kim et al. (2016) reuses these datasets and adds Arabic. Ling et al. (2015) evaluate on English, Portuguese, Catalan, German and Turkish datasets extracted from Wikipedia. Verwimp et al. (2017) use a subset of the Corpus of Spoken Dutch (Oostdijk, 2000) for Dutch LM. Kawakami et al. (2017) evaluate on 7 European languages using Wikipedia data, including Finnish.

Perhaps the largest and most diverse set of languages used for multilingual LM evaluation so far is the one of Vania and Lopez (2017). Their study includes 10 languages in total representing several morphological types (fusional, e.g., Russian, and agglutinative, e.g., Finnish), as well as languages with particular morphological phenomena (root-and-pattern in Hebrew and reduplication in Malay). In this work, we provide LM evaluation datasets for 50 typologically diverse languages, with their selection guided by structural properties.

Language Selection Aiming for a comprehensive multilingual LM evaluation, we include languages for all possible types of morphological systems. Our starting point is the Polyglot Wikipedia (PW) (Al-Rfou et al., 2013). While at first PW seems comprehensive and quite large already (covering 40 languages), the majority of the PW languages are similar from both a genealogical perspective (26/40 are Indo-European) and a geographic perspective (28/40 Western European). As a consequence, they share many patterns and are not a representative sample of the world's languages.

In order to quantitatively analyse global trends and cross-linguistic generalisations across a large set of languages, we propose to test on all PW languages and source additional data from the same domain, Wikipedia[1], considering candidates in descending order of corpus size and morphological type. Traditionally, languages have been grouped into the four

[1]Chinese, Japanese, and Thai are sourced from Wikipedia and processed with the Polyglot tokeniser since we found their preprocessing in the PW is not adequate for language modeling.

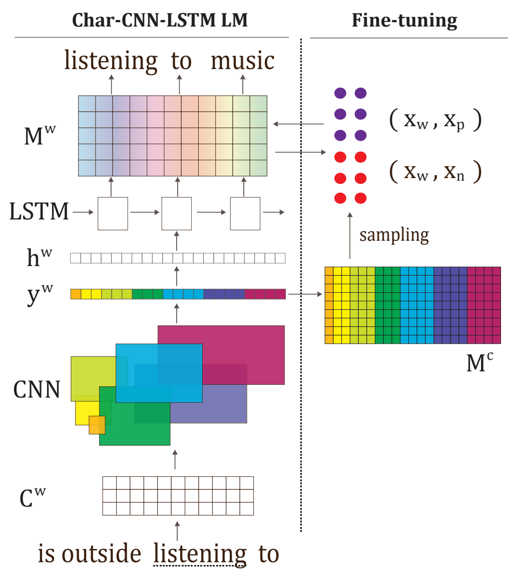

Figure 1: An illustration of the Char-CNN-LSTM LM and our fine-tuning post-processing method. After each epoch we adapt word-level vectors in the softmax embedding M^w using samples based on features from the char-level convolutional filters. The figure follows the model flow bottom to the top.

main types: *isolating*, *fusional*, *introflexive* and *agglutinative*, based on their position along a spectrum measuring their preference on breaking up concepts in many words (on one extreme) or rather compose them into single words (on the other extreme).

However, even languages belonging to the same type display different out-of-vocabulary rates and type-token ratios. This happens because languages specify different subsets of grammatical categories (such as tense for verbs, or number for nouns) and values (such as future for tense, plural for number). The amount of grammatical categories expressed in a language determines its *inflectional synthesis* (Bickel and Nichols, 2013).

In our final sample of languages, we select languages belonging to morphological types different from the fusional one, which is over-represented in the PW. In particular, we include new isolating (*Min Nan, Burmese, Khmer*), agglutinative (*Basque, Georgian, Kannada, Tamil, Mongolian, Javanese*), and introflexive languages (*Amharic*).

3 Underlying LM: Char-CNN-LSTM

As the underlying model we opt for the state-of-the-art neural LM architecture of Kim et al. (2016): it has been shown to work across a number of languages and in a large-scale setup (Jozefowicz et al., 2016). It already provides a solution for the *input* side parameters of the model by building word vectors based on the word's constituent character sequences. However, its *output* side still operates with a standard word-level matrix within the closed and limited vocabulary assumption. We refer to this model as **Char-CNN-LSTM** and describe its details in the following. Figure 1 (left) illustrates the model architecture.

Char-CNN-LSTM constructs input word vectors based on the characters in each word using a convolutional neural network (CNN) (LeCun et al., 1989), then processes the input word-level using a LSTM (Hochreiter and Schmidhuber, 1997). The next word is predicted using word embeddings, a large number of parameters which have to be trained specifically to represent the semantics of single words. We refer to this space of word representations as M^w.

Formally, for the input layer the model trains a look-up matrix $C \in \mathbb{R}^{|V^c| \times d_c}$, corresponding to one d_c-dimensional vector per character c in the char vocabulary V^c. For each input, it takes a sequence of characters of a fixed length m, $[c_1, ...c_m]$, where m is the maximum length of all words in the word vocabulary V^w, and the length of each word is $l \leq m$.

Looking up all characters of a word yields a sequence of char representations in $\mathbb{R}^{d_c \times l}$, which is zero-padded to fit the fixed length m. For each word one gets a sequence of char representations $C^w \in \mathbb{R}^{d_c \times m}$, passed through a 1D convolution:

$$f_i^w = tanh(\langle C^w, H_i \rangle + b). \tag{2}$$

$H_i \in \mathbb{R}^{d_{f,i} \times s_i}$ is a *filter* or *kernel* of size/width s_i and $\langle A, B \rangle = Tr(AB^T)$ is the Frobenius inner product. The model has multiple filters, H_i, with kernels of different width, s_i, and dimensionality $d_{f,i}$, i is used to index filters. Since the model performs a convolution over char embeddings, s_i corresponds to the char window the convolution is operating on: e.g., a filter of width $s_i = 3$ and $d_{3,i} = 150$ could be seen as learning 150 features for detecting 3-grams.

By learning kernels of different width, s_i, the model can learn subword-level features for character sequences of different lengths. f_i^w is the output of taking the convolution with filter H_i for word w. Since f_i^w can get quite large, its dimensionality is reduced using *max-over-time* (1D) pooling: $y_i^w = \max_j f_i^w[j]$. Here, j indexes the dimensions $d_{f,i}$ of the filter f_i^w, and $y_i^w \in \mathbb{R}^{d_{f,i}}$. This corresponds to taking the maximum value for each feature of H_i, with the intuition that the most informative feature would have the highest activation. The output of all max-pooling operations y_i^w is concatenated to form a word vector $y^w \in \mathbb{R}^{d_p}$, where d_p is the number of all features for all H_i:

$$y^w = concat([y_1^w, ..., y_i^w]). \tag{3}$$

This vector is passed through a highway network (Srivastava et al., 2015) to give the network the possibility to reweigh or transform the features: $h^w = Highway(y^w)$. So far all transformations were done per word; after the highway transformation word representations are processed in a sequence by an LSTM (Hochreiter and Schmidhuber, 1997):

$$o_t^w = LSTM([h_{w_1}, ...h_{w_{t-1}}]). \tag{4}$$

The LSTM yields one output vector o_t^w per word in the sequence, given all previous time steps $[y_{w_1}, ...y_{w_{t-1}}]$. To predict the next word w_{t+1}, one takes the dot product of the vector $o_t^w \in \mathbb{R}^{1 \times d_l}$ with a lookup matrix $M^w \in \mathbb{R}^{d_l \times |V^w|}$, where d_l corresponds to the LSTM hidden state size. The vector $p_{t+1} \in \mathbb{R}^{1 \times |V^w|}$ is normalised to contain values between 0 and 1, representing a probability distribution over the next word. This corresponds to calculating the softmax function for every word k in V^w:

$$p(w_{t+1} = k | o_t) = \frac{e^{(o_t \cdot m_k)}}{\sum_{k' \in V_w} e^{(o_t \cdot m_{k'})}} \tag{5}$$

where $P(w_{t+1} = k | o_t)$ is the probability of the next word w_{t+1} being k given o_t, and m_k is the output embedding vector taken from M^w.

Word-Level Vector Space: M^w The model parameters in M^w can be seen as the bottleneck of the model, as they need to be trained specifically for single words, leading to unreliable estimates for infrequent words. As an analysis of the corpus statistics later in Section 7 reveals, the Zipfian effect and its influence on word vector estimation cannot be fully resolved even with a large corpus, especially taking into account how flexible MRLs are in terms of word formation and combination. Yet, having a good estimate for the parameters in M^w is essential for the final LM performance, as they are directly responsible for the next-word prediction.

Therefore, our aim is to improve the quality of representations in M^w, focusing on infrequent words. To achieve this, we turn to another source of information: character patterns. In other words, since M^w does not have any information about character patterns from lower layers, we seek a way to: **a)** detect words with similar subword structures (i.e., “morpheme”-level information), and **b)** let these words share their semantic information.

4 Character-Aware Vector Space

The CNN part of Char-CNN-LSTM, see Eq. (3), in fact provides information about such subword-level patterns: the model constructs a word vector y^w on-the-fly based on the word’s constituent characters. We let the model construct y^w for all words in the vocabulary, resulting in a *character-aware word vector space* $M^c \in \mathbb{R}^{|V^w| \times d_p}$. The construction of the space is completely unsupervised and independent of the word’s context; only the first (CNN) network layers are activated. Our core idea is to leverage this information obtained from M^c to influence the output matrix M^w, and consequently the network prediction, and extend the model to handle unseen words.

We first take a closer look at the character-aware space M^c, and then describe how to improve and expand the semantic space M^w based on the information contained in M^c (Section 5). Each vocabulary entry in M^c encodes character n-gram patterns about the represented word, for $1 \geq n \leq 7$. The n-gram patterns arise through filters of different lengths, and their maximum activation is concatenated to form each individual vector y^w. The matrix M^c is of dimensionality $|V^w| \times 1100$, where each of the 1,100 dimensions corresponds to the activation of one kernel feature. In practice, dimensions $[0, 1, .. : 50]$ correspond to single-character features, $[50 : 150]$ to character 2-grams, $[150 : 300]$ to 3-grams. The higher-order n-grams get assigned 200 dimensions each, up to dimensions $[900 : 1100]$ for 7-grams.

Drawing an analogy to work in computer vision (Zeiler and Fergus, 2014; Chatfield et al., 2014), we delve deeper into the filter activations and analyse the key properties of the vector space M^c. The qualitative analysis reveals that many features are interpretable by humans, and indeed correspond to frequent subword patterns, as illustrated in Table 1. For instance, tokenised Chinese data favours short words: consequently short filters activate strongly for one or two characters. The first two filters (width 1) are highly active for two common single characters each: one filter is active for 更 *(again, more)*, 不 *(not)*, and the other for 今 *(now)*, 代 *(time period)*. Larger filters (width 5-7) do not show interpretable patterns in Chinese, since the vocabulary largely consists of short words (length 1-4).

	s_i	Pattern	Max Activations
ZH	1	更, 不	更为, 更改, 更名, .., 不满, 不明, 不易
	1	今, 代	今日, 今人, 至少, .., 如何, 现代, 当代
TR	1	Caps	**I**n, **E**bru, **VIC**,..,**FAT**, **MW**, **MIT**
	3	mu-	.., **mu**tfağının, **mu**harebe, **mu**htelif
	6	Üniversite	.., **Üniversite**si’nin, **üniversite**lerde

Table 1: Each CNN filter tends to have high activations for a small number of subword patterns. s_i denotes the filter size.

	Word	Nearest Neighbours
DE	Ursprünglichkeit	**ursprünglich**e, **Ur**stoff, **ursprünglich**en
	Mittelwert	**Mittelwert**en,Regel**werk**es,**Mittel**weser
	effektiv	**Effekt**,Perf**ekt**,**Effekt**e,per**fekt**en,Resp**ekt**
JA	大学	大金, 大石, 大震災, 大空, 大野
	ハイク	ハイム , バイク, メイク, ハッサク
	1725	**1**8**25**, **1**6**25**, **152**4mm, **172**8
	Magenta	**Ma**plet, **Ma**ya, **Ma**nagement

Table 2: Nearest neighbours for vocabulary words, based on the character-aware vector space M^c.

Agglutinative languages show a tendency towards long words. We find that medium-sized filters (width 3-5) are active for morphemes or short common subword units, and the long filters are activated for different surface realisations of the same root word. In Turkish, one filter is highly active on various forms of the word *üniversite (university)*. Further, in MRLs with the Latin alphabet short filters are typically active on capitalisation or special chars.

Table 2 shows examples of nearest neighbours based on the activations in M^c. The space seems to be arranged according to shared subword patterns based on the CNN features. It does not rely only on a simple character overlap, but also captures shared morphemes. This property is exploited to influence the LM output word embedding matrix M^w in a completely unsupervised way, as illustrated on the right side of Figure 1.

5 Fine-Tuning the LM Prediction

While the output vector space M^w captures word-level semantics, M^c arranges words by subword features. A model which relies solely on character-level knowledge (similar to the information stored in M^c) for word-level prediction cannot fully capture word-level semantics and even hurts LM performance (Jozefowicz et al., 2016). However, shared subword units still provide useful evidence of shared semantics (Cotterell et al., 2016; Vulić et al., 2017): injecting this into the space M^w to *additionally* reflect shared subword-level information should lead to improved word vector estimates, especially for MRLs.

5.1 Fine-Tuning and Constraints

We inject this information into M^w by adapting recent fine-tuning (often termed *retrofitting* or *specialisation*) methods for vector space post-processing (Faruqui et al., 2015; Wieting et al., 2015; Mrkšić et al., 2017; Vulić et al., 2017, i.a.). These models enrich initial vector spaces by encoding external knowledge provided in the form of simple *linguistic constraints* (i.e., word pairs) into the initial vector space.

There are two fundamental differences between our work and previous work on specialisation. First, previous models typically use rich hand-crafted lexical resources such as WordNet (Fellbaum, 1998) or the Paraphrase Database (Ganitkevitch et al., 2013), or manually defined rules (Vulić et al., 2017) to extract the constraints, while we generate them directly using the implicit knowledge coded in M^c. Second, our method is *integrated* into a language model: it performs updates after each epoch of the LM training.[2] In Section 5.2, we describe our model for fine-tuning M^w based on the information provided in M^c.

Our fine-tuning approach relies on constraints: positive and negative word pairs (x_i, x_j), where $x_i, x_j \in V^w$. Iterating over each *cue word* $x_w \in V^w$ we find a set of positive word pairs P_w and negative word pairs N_w: their extraction is based on their (dis)similarity with x_w in M^c. Positive pairs (x_w, x_p) contain words x_p yielding the highest cosine similarity to the x_w (=nearest neighbors) in M^c. Negative pairs (x_w, x_n) are constructed by randomly sampling words x_n from the vocabulary. Since M^c gets updated during the LM training, we (re)generate the sets P_w and N_w after each epoch.

[2]We have also experimented with a variant which performs only a post-hoc single update of the M^w matrix after the LM training, but a variant which performs continuous per-epoch updates is more beneficial for the final LM performance.

5.2 Attract-Preserve

We now present a method for fine-tuning the output matrix M^w within the Char-CNN-LSTM LM framework. As said, the fine-tuning procedure runs after each epoch of the standard log-likelihood LM training (see Figure 1). We adapt a variant of a state-of-the-art post-processing specialisation procedure (Wieting et al., 2015; Mrkšić et al., 2017). The idea of the fine-tuning method, which we label **Attract-Preserve (AP)**, is to pull the positive pairs closer together in the output word-level space, while pushing the negative pairs further away.

Let v_i denote the word vector of the word x_i. The AP cost function has two parts: *attract* and *preserve*. In the *attract* term, using the extracted sets P_w and N_w, we push the vector of x_w to be closer to x_p by a similarity margin δ than to its negative sample x_n:

$$attr(P_w, N_w) = \sum_{\substack{(x_w,x_p)\in P_w,\\(x_w,x_n)\in N_w}} ReLU(\delta + v_w v_n - v_w v_p).$$

$ReLU(x)$ is the standard rectified linear unit (Nair and Hinton, 2010). The δ margin is set to 0.6 in all experiments as in prior work (Mrkšić et al., 2017) without any subsequent fine-tuning.

The preserve cost acts as a regularisation pulling the "fine-tuned" vector back to its initial value:

$$pres(P_w, N_w) = \sum_{x_w \in V^w} \lambda_{reg} ||\hat{v}_w - v_w||_2. \quad (6)$$

$\lambda_{reg} = 10^{-9}$ is the L_2-regularisation constant (Mrkšić et al., 2017); $\hat{v}_w$ is the original word vector before the procedure. This term tries to preserve the semantic content present in the original vector space, as long as this information does not contradict the knowledge injected by the constraints. The final cost function adds the two costs: $cost = attr + pres$.

6 Experiments

Datasets We use the Polyglot Wikipedia (Al-Rfou et al., 2013) for all available languages except for Japanese, Chinese, and Thai, and add these and further languages using Wikipedia dumps. The Wiki dumps were cleaned and preprocessed by the Polyglot tokeniser. We construct similarly-sized datasets by extracting 46K sentences for each language from the beginning of each dump, filtered to contain only full sentences, and split into train (40K), validation (3K), and test (3K). The final list of languages along with standard language codes (ISO 639-1 standard, used throughout the paper) and statistics on vocabulary and token counts are provided in Table 4.

Evaluation Setup We report *perplexity* scores (Jurafsky and Martin, 2017, Chapter 4.2.1) using the *full* vocabulary of the respective LM dataset. This means that we explicitly decide to retain also infrequent words in the modeled data. Replacing infrequent words by a placeholder token <*unk*> is a standard technique in LM to obtain equal vocabulary sizes across different datasets. Motivated by the observation that infrequent words constitute a significant part of the vocabulary in MRLs, and that vocabulary sizes naturally differ between languages, we have decided to avoid the <*unk*> placeholder for low-frequency words, and run all models on the full vocabulary (Adams et al., 2017; Grave et al., 2017).

We believe that this full-vocabulary setup offers additional insight into the standard LM techniques, leading to evaluation which pinpoints crucial limitations of current word-based models with regard to morphologically-rich languages. In our setup the vocabulary contains all words occurring at least once in the training set. To ensure a fair comparison between all neural models, words occurring only in the test set are mapped to a random vector with the same technique for all neural models, as described next.

Sampling Vectors of Unseen Words Since zero-shot semantic vector estimation at test time is an unresolved problem, we seek an alternative way to compare model predictions at test time. We report all results with unseen test words being mapped to *one* randomly sampled <*unk*> vector. The <*unk*> vector is part of the vocabulary at training time, but remains untrained and at its random initialization since it never occurs in the training data. Therefore, we sample a random <*unk*> vector at test time from the same part of the space as the trained vectors, using a normal distribution with the mean and the variance of M^w and the same fixed random seed for all models. We employ this methodology for *all* neural LM models, and thereby ensure that results are comparable.

Character embedding size	15
Word embedding size	650
Number of RNN layers	2
Number of highway layers	2
Dropout value	0.5
Optimizer	SGD
Learning rate	1.0
Learning rate decay	0.5
Parameter init: rand uniform	[-0.05, 0.05]
Batch size	20
RNN sequence length	35
Max grad norm	5.0
Max word length	dynamic
Max epochs	15 or 30
AP margin (δ)	0.6
AP optimizer	Adagrad
AP learning rate	0.05
AP gradient clip	2
AP regularization constant	10^{-9}
AP rare words frequency threshold	5

Table 3: Hyper-parameters.

Training Setup and Parameters We reproduce the standard LM setup of Zaremba et al. (2015) and parameter choices of Kim et al. (2016), with batches of 20 and a sequence length of 35, where one step corresponds to one token. The maximum word length is chosen dynamically based on the longest word in the corpus. The corpus is processed continuously, and the RNN hidden state resets occur at the beginning of each epoch. Parameters are optimised with stochastic gradient descent. The gradient is averaged over the batch size and sequence length. We then scale the averaged gradient by the sequence length (=35) and clip to 5.0 for more stable training. The learning rate is 1.0, decayed by 0.5 after each epoch if the validation perplexity does not improve. We train all models for 15 epochs on the smaller corpora, and for 30 on the large ones, which is typically sufficient for model convergence.

Our AP fine-tuning method operates on the whole

M^w space, but we only allow words more frequent than 5 as cue words x_w (see Section 5 again), while there are no restrictions on x_p and x_n.[3] Our preliminary analysis on the influence of the number of nearest neighbours in M^c shows that this parameter has only a moderate effect on the final LM scores. We thus fix it to 3 positive and 3 negative samples for each x_w without any tuning. AP is optimised with Adagrad (Duchi et al., 2011) and a learning rate of 0.05, the gradients are clipped to ± 2.[4] A full summary of all hyper-parameters and their values is provided in Table 3.

(Baseline) Language Models The availability of LM evaluation sets in a large number of diverse languages, described in Section 2, now provides an opportunity to perform a full-fledged multilingual analysis of representative LM architectures. At the same time, these different architectures serve as the baselines for our novel model which fine-tunes the output matrix M^w.

As mentioned, the traditional LM setup is to use words both on the input and on the output side (Goodman, 2001; Bengio et al., 2003; Deschacht and Moens, 2009) relying on n-gram word sequences. We evaluate a strong model from the *n-gram* family of models from the KenLM package (*https://github.com/kpu/kenlm*): it is based on 5-grams with extended Kneser-Ney smoothing (**KN5**) (Kneser and Ney, 1995; Heafield et al., 2013) [5]. The rationale behind including this *non-neural* model is to also probe the limitations of such n-gram-based LM architectures on a diverse set of languages.

Recurrent neural networks (RNNs), especially Long-Short-Term Memory networks (LSTMs), have taken over the LM universe recently (Mikolov et al., 2010; Sundermeyer et al., 2015; Chen et al., 2016, i.a.). These LMs map a sequence of input words to embedding vectors using a look-up matrix. The embeddings are passed to the LSTM as input, and the model is trained in an autoregressive fashion to predict the next word from the pre-defined vocabulary given the current context. As a strong baseline from this LM family, we train a standard LSTM LM (**LSTM-Word**) relying on the setup from Zaremba et al. (2015) (see Table 3).

Finally, a recent strand of LM work uses characters on the input side while retaining word-level prediction on the output side. A representative architecture from this group, also serving as the basis in our work (Section 3), is **Char CNN-LSTM** (Kim et al., 2016).

All neural models operate on exactly the same vocabulary and treat out-of-vocabulary (OOV) words in exactly the same way. As mentioned, we include KN5 as a strong (non-neural) baseline to give perspective on how this more traditional model performs across 50 typologically diverse languages. We have selected the setup for the KN5 model to be as close as possible to that of neural LMs, However, due to the different nature of the models, we note that the results between KN5 and other models are not comparable.

In KN5 discounts are added for low-frequency words, and unseen words at test time are regarded as outliers and assigned low probability estimates. In contrast, for all neural models we sample unseen word vectors to lie in the space of trained vectors (see before). We find the latter setup to better reflect our intuition that especially in MRLs unseen words are not outliers but often arise due to morphological complexity.

7 Results and Discussion

In this section, we present the main empirical findings of our work. The focus is on: **a)** the results of our novel language model with the AP fine-tuning procedure, and its comparison to other language models in our comparison; **b)** the analysis of the LM results in relation to typological features and corpus statistics.

Table 4 that lists all 50 test languages along with their language codes and provides the key statistics of our 50 LM evaluation benchmarks. The statistics include the number of word types in training data, the number of word types occurring in test data but unseen in training, as well as the total number of word tokens in both training and test data, and type-to-token ratios.

Table 4 also shows the results for KN5, LSTM-

[3]This choice has been motivated by the observation that rare words tend to have other rare words as their nearest neighbours. Note that vectors of words from positive and negative examples, and not only cue words, also get updated by the AP method.

[4]All scores with neural models are produced with our own implementations in TensorFlow (Abadi et al., 2016).

[5]We evaluate the default setup for this model using the option `-interpolate_unigrams=1` which avoids assigning zero-probability to unseen words.

	Data Stats					**Baseline Models**			**Ours: Fine-Tuning M^w**	
Language (code)	Vocab Size (Train)	New Test Vocab	Number Tokens (Train)	Number Tokens (Test)	Type / Token (Train)	KN5	LSTM	Char-CNN-LSTM	+AP	Δ +AP
× Amharic (am)	89749	4805	511K	39.2K	0.18	1252	1535	981	**817**	164
× Arabic (ar)	89089	5032	722K	54.7K	0.12	2156	2587	1659	**1604**	55
☐ Bulgarian (bg)	71360	3896	670K	49K	0.11	610	651	415	**409**	6
☐ Catalan (ca)	61033	2562	788K	59.4K	0.08	358	318	241	**238**	3
☐ Czech (cs)	86783	4300	641K	49.6K	0.14	1658	2200	1252	**1131**	121
☐ Danish (da)	72468	3618	663K	50.3K	0.11	668	710	466	**442**	24
☐ German (de)	80741	4045	682K	51.3K	0.12	930	903	602	**551**	51
☐ Greek (el)	76264	3767	744K	56.5K	0.10	607	538	405	**389**	16
☐ English (en)	55521	2480	783K	59.5K	0.07	533	494	371	**349**	22
☐ Spanish (es)	60196	2721	781K	57.2K	0.08	415	366	275	**270**	5
★ Estonian (et)	94184	3907	556K	38.6K	0.17	1609	2564	1478	**1388**	90
★ Basque (eu)	81177	3365	647K	47.3K	0.13	560	533	347	**309**	38
☐ Farsi (fa)	52306	2041	738K	54.2K	0.07	355	263	208	**205**	3
★ Finnish (fi)	115579	6489	585K	44.8K	0.20	2611	4263	2236	**1858**	378
☐ French (fr)	58539	2575	769K	57.1K	0.08	350	294	231	**220**	11
× Hebrew (he)	83217	3862	717K	54.6K	0.12	1797	2189	1519	**1375**	144
☐ Hindi (hi)	50384	2629	666K	49.1K	0.08	473	426	326	**299**	27
☐ Croatian (hr)	86357	4371	620K	48.1K	0.14	1294	1665	1014	**906**	108
★ Hungarian (hu)	101874	5015	672K	48.7K	0.15	1151	1595	929	**819**	110
▷ Indonesian (id)	49125	2235	702K	52.2K	0.07	454	359	286	**263**	23
☐ Italian (it)	70194	2923	787K	59.3K	0.09	567	493	**349**	350	-1
★ Japanese (ja)	44863	1768	729K	54.6K	0.06	169	156	136	**125**	11
★ Javanese (jv)	65141	4292	622K	52K	0.10	1387	1443	1158	**1003**	155
★ Georgian (ka)	80211	3738	580K	41.1K	0.14	1370	1827	1097	**939**	158
▷ Khmer (km)	37851	1303	579K	37.4K	0.07	586	637	**522**	535	-13
★ Kannada (kn)	94660	4604	434K	29.4K	0.22	2315	5310	2558	**2265**	293
★ Korean (ko)	143794	8275	648K	50.6K	0.22	5146	10063	4778	**3821**	957
☐ Lithuanian (lt)	81501	3791	554K	41.7K	0.15	1155	1415	854	**827**	27
☐ Latvian (lv)	75294	4564	587K	45K	0.13	1452	1967	1129	**969**	160
▷ Malay (ms)	49385	2824	702K	54.1K	0.07	776	725	525	**513**	12
★ Mongolian (mng)	73884	4171	629K	50K	0.12	1392	1716	1165	**1091**	74
▷ Burmese (my)	20574	755	576K	46.1K	0.04	209	212	182	**180**	2
▷ Min-Nan (nan)	33238	1404	1.2M	65.6K	0.03	61	43	39	38	1
☐ Dutch (nl)	60206	2626	708K	53.8K	0.08	397	340	267	**248**	19
☐ Norwegian (no)	69761	3352	674K	47.8K	0.10	534	513	379	**346**	33
☐ Polish (pl)	97325	4526	634K	47.7K	0.15	1741	2641	1491	**1328**	163
☐ Portuguese (pt)	56167	2394	780K	59.3K	0.07	342	272	214	**202**	12
☐ Romanian (ro)	68913	3079	743K	52.5K	0.09	384	359	256	**247**	9
☐ Russian (ru)	98097	3987	666K	48.4K	0.15	1128	1309	812	**715**	97
☐ Slovak (sk)	88726	4521	618K	45K	0.14	1560	2062	1275	**1151**	124
☐ Slovene (sl)	83997	4343	659K	49.2K	0.13	1114	1308	776	**733**	43
☐ Serbian (sr)	81617	3641	628K	46.7K	0.13	790	961	582	**547**	35
☐ Swedish (sv)	77499	4109	688K	50.4K	0.11	843	832	583	**543**	40
★ Tamil (ta)	106403	6017	507K	39.6K	0.21	3342	6234	3496	**2768**	728
▷ Thai (th)	30056	1300	628K	49K	0.05	233	241	206	**199**	7
▷ Tagalog (tl)	72416	3791	972K	66.3K	0.07	379	298	219	**211**	8
★ Turkish (tr)	90840	4608	627K	45K	0.14	1724	2267	1350	**1290**	60
☐ Ukranian (uk)	89724	4983	635K	47K	0.14	1639	1893	1283	**1091**	192
▷ Vietnamese (vi)	32055	1160	754K	61.9K	0.04	197	190	**158**	165	-7
▷ Chinese (zh)	43672	1653	746K	56.8K	0.06	1064	826	797	**762**	35
▷ **Isolating** (avg)	40930	1825	759K	54K	0.05	440	392	326	**318**	8
☐ **Fusional** (avg)	73499	3532	689K	51.3K	0.11	842	969	618	**566**	52
× **Introflexive** (avg)	87352	4566	650K	49.5K	0.14	1735	2104	1386	**1265**	121
★ **Agglutinative** (avg)	91051	4687	603K	45K	0.16	1898	3164	1727	**1473**	254

Table 4: Test perplexities for 50 languages (ISO 639-1 codes sorted alphabetically) in the full-vocabulary prediction LM setup; **Left**: Basic statistics of our evaluation data. **Middle**: Results with the *Baseline LMs*. Note that the absolute scores in the KN5 column are not comparable to the scores obtained with neural models (see Section 6). **Right**: Results with Char-CNN-LSTM and our AP fine-tuning strategy. Δ is indicating the difference in performance over the original Char-CNN-LSTM model. The best scoring neural baseline is underlined. The overall best performing neural model for each language is in bold.

Word, Char-CNN-LSTM, and our model with the AP fine-tuning. Furthermore, a visualisation of the Char-CNN-LSTM+AP model as a function of type/token ratio is shown in Figure 2.

7.1 Fine-Tuning the Output Matrix

First, we test the impact of our AP fine-tuning method. As the main finding, the inclusion of fine-tuning into Char-CNN-LSTM (this model is termed

+AP) yields improvements on a large number of test languages. The model is better than both strong neural baseline language models for 47/50 languages, and it improves over the original Char-CNN-LSTM LM for 47/50 languages. The largest gains are indicated for the subset of agglutinative MRLs (e.g., 950 perplexity points in Korean, large gains also marked for FI, HE, KA, HU, TA, ET). We also observe large gains for the three introflexive languages included in our study (Amharic, Arabic, Hebrew).

While these large absolute gains may be partially attributed to the exponential nature of the perplexity measure, one cannot ignore the substantial relative gains achieved by our models: e.g., EU (ΔPPL=38) improves more than a fusional language like DA (ΔPPL=24) even with a lower baseline perplexity. This suggests that injecting subword-level information is more straightforward for the former: in agglutinative languages, the mapping between morphemes and meanings is less ambiguous. Moreover, the number of words that benefit from the injection of character-based information is larger for agglutinative languages, because they also tend to display the highest inflectional synthesis.

For the opposite reasons, we do not surpass Char-CNN-LSTM in a few fusional (IT) and isolating languages (KM, VI). We also observe improvements for Slavic languages with rich morphology (RU, HR, PL). The gains are also achieved for some isolating and fusional languages with smaller vocabularies and a smaller number of rare words, e.g., in Tagalog, English, Catalan, and Swedish. This suggests that our method for fine-tuning the LM prediction is not restricted to MRLs only, and has the ability to improve the estimation for rare words in multiple typologically diverse languages.

7.2 Language Models, Typological Features, and Corpus Statistics

In the next experiment, we estimate correlation strength of all perplexity scores with a series of independent variables. The variables are 1) type-token ratio in the train data; 2) new word types in the test data; 3) the morphological type of the language among *isolating*, *fusional*, *introflexive*, and *agglutinative*, capturing different aspects related to the morphological richness of a language.

Results with Pearson's ρ (numerical) and η^2 in

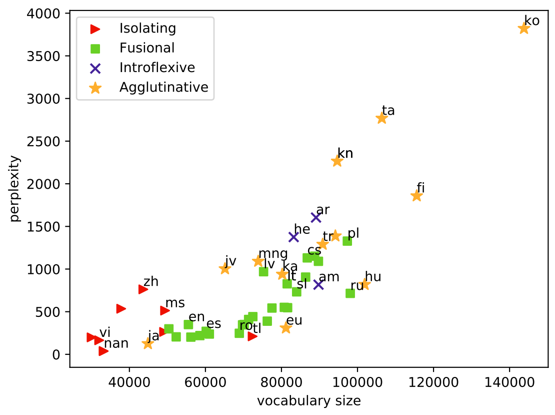

Figure 2: Perplexity results with Char-CNN-LSTM+AP (y-axis) in relation to type/token ratio (x-axis). For language codes, see Table 4.

one-way ANOVA (categorical) are shown in Table 5. Significance tests show p-values $< 1^{-3}$ for all combinations of models and independent variables, demonstrating all of them are good performance predictors. Our main finding indicates that linguistic categories and data statistics both correlate well (≈ 0.35 and ≈ 0.82, respectively) with the performance of language models.

For the categorical variables we compare the mean values per category with the numerical dependent variable. As such, η^2 can be interpreted as the amount of variation explained by the model - the resulting high correlations suggest that perplexities tend to be homogeneous for languages of a same morphological type, especially so for state-of-the-art models.

This is intuitively evident in Figure 2, where perplexity scores of Char-CNN-LSTM+AP are plotted against type/token ratio. Isolating languages are placed on the left side of the spectrum as expected, with low type/token ratio and good performance (e.g., VI, ZH). As for fusional languages, sub-groups behave differently. We find that Romance and Germanic languages display roughly the same level of performance as isolating languages, despite their overall larger type/token ratio. Balto-Slavic languages (e.g. CS, LV) instead show both higher perplexities and higher type/token ratio. These differences may be explained in terms of different inflectional synthesis.

Introflexive and agglutinative languages can be

Variables						
Independent	Dependent	Statistical Test	Models			
			KN5	LSTM	+Char-CNN	++AP
Train type/token	PPL	Pearson's ρ 0.833	0.813	0.823	0.831	
Test new types	PPL	Pearson's ρ	0.860	0.803	0.818	0.819
Morphology	PPL	one-way ANOVA η^2	0.354	0.338	0.369	0.374
			LSTM vs +CharCNN		+CharCNN vs ++AP	
Train type/token	Δ PPL	Pearson's ρ	0.729		0.778	
Morphology	Δ PPL	one-way ANOVA η^2	0.308		0.284	

Table 5: Correlations between model performance and language typology as well as with corpus statistics (type/token ratio and new word types in test data). All variables are good performance predictions.

found mostly on the right side of the spectrum in terms of performance (see Figure 2). Although the languages with highest absolute perplexity scores are certainly classified as agglutinative (e.g., Dravidian languages such as KN and TA), we also find some outliers in the agglutinative languages (EU) with remarkably low perplexity scores.

7.3 Corpus Size and Type/Token Ratio

Building on the strong correlation between type/token ratio and model performance from Section 7.2, we now further analyse the results in light of corpus size and type/token statistics. The LM datasets for our 50 languages are similar in size to the widely used English PTB dataset (Marcus et al., 1993). As such, we hope that these evaluation datasets can help guide multilingual language modeling research across a wide spectrum of languages.

However, our goal now is to verify that type/token ratio and not absolute corpus size is the deciding factor when unraveling the limitations of standard LM architectures across different languages. To this end, we conduct additional experiments on all languages of the recent Multilingual Wikipedia Corpus (MWC) (Kawakami et al., 2017) for language modeling, using the same setup as before (see Table 3). The corpus provides datasets for 7 languages from the same domain as our benchmarks (Wikipedia), and comes in two sizes. We choose the larger corpus variant for each language, which provides about 3-5 times as many tokens as contained in our data sets from Table 4.

The results on the MWC evaluation data along with corpus statistics are summarised in Table 6. As one important finding, we observe that the gains in perplexity using our fine-tuning AP method extend also to these larger evaluation datasets. In particular, we find improvements of the same magnitude as in the PTB-sized data sets over the strongest baseline model (Char-CNN-LSTM) for all MWC languages. For instance, perplexity is reduced from 1781 to 1578 for Russian, and from 365 to 352 for English. We also observe a gain for French and Spanish with perplexity reduced from 282 to 272 and 255 to 243 respectively.

In addition, we test on samples of the Europarl corpus (Koehn, 2005; Tiedemann, 2012) which contains approximately 10 times more tokens than our PTB-sized evaluation data: we use 400K sentences from Europarl for training and testing. However, this data comes from a much narrower domain of parliamentary proceedings: this property yields a very low type/token ratio as visible from Table 6. In fact, we find the type/token ratio in this corpus to be on the same level or even smaller than isolating languages (compare with the scores in Table 4): 0.02 for Dutch and 0.03 for Czech. This leads to similar perplexities with and without +AP for these two selected test languages. The third EP test language, Finnish, has a slightly higher type/token ratio. Consequently, we do observe an improvement of 10 points in perplexity. A more detailed analysis of this phenomenon follows.

Table 7 displays the overall type/token ratio in the training set of these copora. We observe that the MWC has comparable or even higher type/token ratios than the smaller sets despite its increased size. The corpus has been constructed by sampling the data from a variety of different Wikipedia categories (Kawakami et al., 2017): it can therefore be regarded as more diverse and challenging to model.

Lang	Corpus	# Vocab train	# Vocab test	# Tokens train	# Tokens test	Type/Token train	Char-CNN-LSTM	+AP
nl	EP	197K	200K	10M	255K	0.02	**62**	63
cs	EP	265K	268K	7.9M	193K	0.03	**180**	186
en	MWC	310K	330K	5.0M	0.5M	0.06	365	**352**
es	MWC	258K	277K	3.7M	0.4M	0.07	255	**243**
fr	MWC	260K	278K	4.0M	0.5M	0.07	282	**272**
fi	EP	459K	465K	6.8M	163K	0.07	515	**505**
de	MWC	394K	420K	3.8M	0.3M	0.10	710	**665**
ru	MWC	372K	399K	2.5M	0.3M	0.15	1781	**1578**
cs	MWC	241K	258K	1.5M	0.2M	0.16	2396	**2159**
fi	MWC	320K	343K	1.5M	0.1M	0.21	5300	**4911**

Table 6: Results on the larger MWC data set (Kawakami et al., 2017) and on a subset of the Europarl (EP) corpus. Improvements with +AP are not dependent on corpus size, but rather they strongly correlate with the type/token ratio of the corpus.

Language	Type/Token Ratio Our Data	MWC	Europarl
Czech	0.13	0.16	0.03
German	0.12	0.10	-
English	0.06	0.06	-
Spanish	0.07	0.07	-
Finnish	0.20	0.21	0.07
French	0.07	0.07	-
Russian	0.14	0.15	-
Dutch	0.09	-	0.02

Table 7: Comparison of type/token ratios in the corpora used for evaluation. The ratio is not dependent only on the corpus size but also on the language and domain of the corpus.

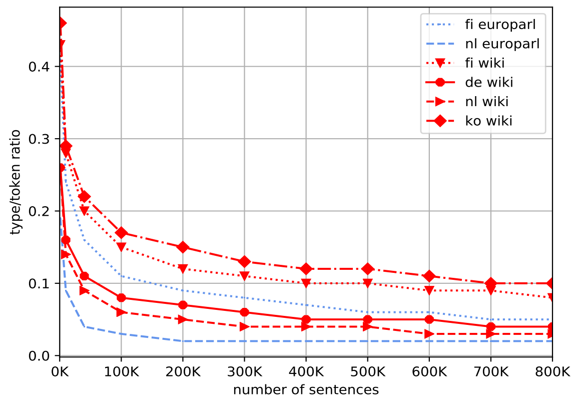

Figure 3: Type/token ratio values vs. corpus size. A domain-specifc corpus (Europarl) has a lower type/token ratio than a more general corpus (Wikipedia), regardless of the absolute corpus size.

Europarl on the other hand shows substantially lower type/token ratios, presumably due to its narrower domain and more repetitive nature.

In general, we find that although the type/token ratio decreases with increasing corpus size, the decreasing rate slows down dramatically at a certain point (Herdan, 1960; Heaps, 1978). This depends on the typology of the language and domain of the corpus. Figure 3 shows the empirical proof of this intuition. We show the variation of type/token ratios in Wikipedia and Europarl with increasing corpus size. We can see that in a very large corpus of 800K sentences, the type/token ratio in MRLs such as Korean or Finnish stays close to 0.1, a level where we still expect an improvement in perplexity with the proposed AP fine-tuning method applied on top of Char-CNN-LSTM.

In order to isolate and verify the effect of the type/token ratio, we now present results on synthetically created data sets where the ratio is controlled explicitly. We experiment with subsets of the German Wikipedia with equal number of sentences (25K)[6], comparable number of tokens, but varying type/token ratio. We generate these controlled data sets by clustering sparse bag-of-words sentence vectors with the k-means algorithm, sampling from different clusters,

[6]We split the data into 20K training, 2.5K validation and 2.5K test sentences

Clusters	# Vocab		# Tokens		Type/Token	Char-CNN-LSTM	+AP
	train	test	train	test	train		
2	48K	52K	382K	47K	0.13	225	**217**
2,4	69K	75K	495K	62K	0.14	454	**420**
2,4,5,9	78K	84K	494K	62K	0.16	605	**547**
5,9	84K	91K	492K	62K	0.17	671	**612**
5	66K	72K	372K	46K	0.18	681	**598**

Table 8: Results on German with data sets of comparable size and increasing type/token ratio.

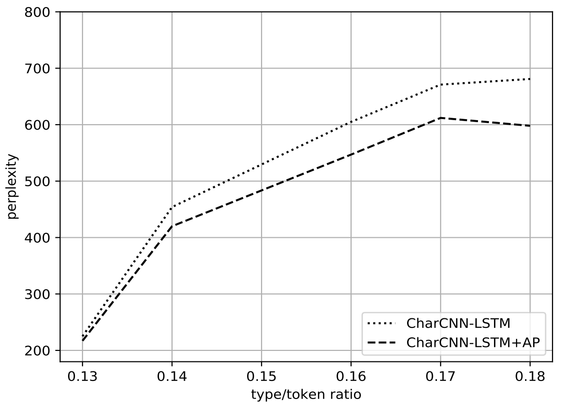

Figure 4: Visualisation of results from Table 8. The AP method is especially helpful for corpora with high type/token ratios.

and then selecting the final combinations according to their type/token ratio and the number of tokens. Corpora statistics along with corresponding perplexity scores are shown in Table 8, and plotted in Figure 4. These results clearly demonstrate and verify that the effectiveness of the AP method increases for corpora with higher type/token ratios. This finding also further supports the usefulness of the proposed method for morphologically-rich languages in particular, where such high type/token ratios are expected.

8 Conclusion

We have presented a comprehensive language modeling study over a set of 50 typologically diverse languages. The languages were carefully selected to represent a wide spectrum of different morphological systems that are found among the world's languages. Our comprehensive study provides new benchmarks and language modeling baselines which should guide the development of next-generation language models focused on the challenging multilingual setting.

One particular LM challenge is an effective learning of parameters for infrequent words, especially for morphologically-rich languages (MRLs). The methodological contribution of this work is a new neural approach which enriches word vectors at the LM output with subword-level information to capture similar character sequences and consequently to facilitate word-level LM prediction. Our method has been implemented as a fine-tuning step which gradually refines word vectors during the LM training, based on subword-level knowledge extracted in an unsupervised manner from character-aware CNN layers. Our approach yields gains for 47/50 languages in the challenging full-vocabulary setup, with largest gains reported for MRLs such as Korean or Finnish. We have also demonstrated that the gains extend to larger training corpora, and are well correlated with the type-to-token ratio in the training data.

In future work we plan to deal with the open vocabulary LM setup and extend our framework to also handle unseen words at test time. One interesting avenue might be to further fine-tune the LM prediction based on additional evidence beyond purely contextual information. In summary, we hope that this article will encourage further research into learning semantic representations for rare and unseen words, and steer further developments in multilingual language modeling across a large number of diverse languages. Code and data are available online: `http://people.ds.cam.ac.uk/dsg40/lmmrl.html`.

Acknowledgments

This work is supported by the ERC Consolidator Grant LEXICAL (648909). We thank all editors and reviewers for their helpful feedback and suggestions.

References

Martin Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Man, Rajat Monga, Sherry Moore, Derek Murray, Jon Shlens, Benoit Steiner, Ilya Sutskever, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Oriol Vinyals, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. TensorFlow: Large-scale machine learning on heterogeneous distributed systems. *CoRR*, abs/1603.04467.

Oliver Adams, Adam Makarucha, Graham Neubig, Steven Bird, and Trevor Cohn. 2017. Cross-lingual word embeddings for low-resource language modeling. In *Proceedings of EACL*, pages 937–947.

Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual NLP. In *Proceedings of CoNLL*, pages 183–192.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

Balthasar Bickel and Johanna Nichols, 2013. *Inflectional Synthesis of the Verb*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Ondřej Bojar, Christian Buck, Chris Callison-Burch, Christian Federmann, Barry Haddow, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2013. Findings of the 2013 Workshop on Statistical Machine Translation. In *Proceedings of the 8th Workshop on Statistical Machine Translation*, pages 1–44.

Jan A. Botha and Phil Blunsom. 2014. Compositional morphology for word representations and language modelling. In *Proceedings of ICML*, pages 1899–1907.

Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2014. Return of the devil in the details: Delving deep into convolutional nets. In *Proceedings of BMVC*.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, and Phillipp Koehn. 2014. One billion word benchmark for measuring progress in statistical language modeling. In *Proceedings of INTERSPEECH*, pages 2635–2639.

Xie Chen, Xunying Liu, Yanmin Qian, M.J.F. Gales, and Philip C Woodland. 2016. CUED-RNNLM: An open-source toolkit for efficient training and evaluation of recurrent neural network language models. In *Proceedings of ICASSP*, pages 6000 –6004.

Ryan Cotterell, Hinrich Schütze, and Jason Eisner. 2016. Morphological smoothing and extrapolation of word embeddings. In *Proceedings of ACL*, pages 1651–1660.

Koen Deschacht and Marie-Francine Moens. 2009. Semi-supervised semantic role labeling using the latent words language model. In *Proceedings of EMNLP*, pages 21–29.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard H Hovy, and Noah A Smith. 2015. Retrofitting Word Vectors to Semantic Lexicons. In *Proceedings of NAACL-HLT*, pages 1606–1615.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. Bradford Books.

Katja Filippova, Enrique Alfonseca, Carlos A. Colmenares, Lukasz Kaiser, and Oriol Vinyals. 2015. Sentence compression by deletion with LSTMs. In *Proceedings of EMNLP*, pages 360–368.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The Paraphrase Database. In *Proceedings of NAACL-HLT*, pages 758–764.

Joshua T. Goodman. 2001. A bit of progress in language modeling. *Computer Speech & Language*, 15(4):403–434.

Edouard Grave, Moustapha Cissé, and Armand Joulin. 2017. Unbounded cache model for online language modeling with open vocabulary. In *Proceedings of NIPS*, pages 6044–6054.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850.

Derek Greene and Padraig Cunningham. 2006. Practical solutions to the problem of diagonal dominance in kernel document clustering. In *Proceedings of ICML*, pages 377–384.

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. 2013. Scalable modified Kneser-Ney language model estimation. In *Proceedings of ACL*, pages 690–696.

Harold Stanley Heaps. 1978. *Information retrieval, computational and theoretical aspects*. Academic Press.

Gustav Herdan. 1960. *Type-token mathematics*, volume 4. Mouton.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Marcus Hutter. 2012. The human knowledge compression contest.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. In *Proceedings of ICML*.

Dan Jurafsky and James H. Martin. 2017. *Speech and Language Processing*, volume 3. Pearson.

Kazuya Kawakami, Chris Dyer, and Phil Blunsom. 2017. Learning to create and reuse words in open-vocabulary neural language modeling. In *Proceedings of ACL*, pages 1492–1502.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2016. Character-aware neural language models. In *Proceedings of AAAI*, pages 2741–2749.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for M-gram language modeling. In *Proceedings of ICASSP*, pages 181–184.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of the 10th Machine Translation Summit*, pages 79–86.

Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne E. Hubbard, and Lawrence D. Jackel. 1989. Handwritten digit recognition with a back-propagation network. In *Proceedings of NIPS*, pages 396–404.

Wang Ling, Tiago Luís, Luís Marujo, Ramón Fernández Astudillo, Silvio Amir, Chris Dyer, Alan W. Black, and Isabel Trancoso. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of EMNLP*, pages 1520–1530.

Minh-Thang Luong and Christopher D. Manning. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. In *Proceedings of ACL*, pages 1054–1063.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of ACL*, pages 142–150.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proceedings of INTERSPEECH*, pages 1045–1048.

Yasumasa Miyamoto and Kyunghyun Cho. 2016. Gated word-character recurrent language model. In *Proceedings of EMNLP*, pages 1992–1997.

Nikola Mrkšić, Ivan Vulić, Diarmuid Ó Séaghdha, Ira Leviant, Roi Reichart, Milica Gašić, Anna Korhonen, and Steve Young. 2017. Semantic specialisation of distributional word vector spaces using monolingual and cross-lingual constraints. *Transactions of the ACL*, 5:309–324.

Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted Boltzmann machines. In *Proceedings of ICML*, pages 807–814.

Nelleke Oostdijk. 2000. The spoken Dutch corpus. Overview and first evaluation. In *Proceedings of LREC*, pages 887–894.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proceedings of EACL*, pages 157–163.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C. Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of AAAI*, pages 3776–3784.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. In *Proceedings of the ICML Deep Learning Workshop*.

Martin Sundermeyer, Hermann Ney, and Ralf Schluter. 2015. From feedforward to recurrent LSTM neural networks for language modeling. *IEEE Transactions on Audio, Speech and Language Processing*, 23(3):517–529.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in OPUS. In *Proceedings of LREC*, pages 2214–2218.

Clara Vania and Adam Lopez. 2017. From characters to words to in between: Do we capture morphology? In *Proceedings of ACL*, pages 2016–2027.

Ashish Vaswani, Yinggong Zhao, Victoria Fossum, and David Chiang. 2013. Decoding with large-scale neural language models improves translation. In *Proceedings of EMNLP*, pages 1387–1392.

Lyan Verwimp, Joris Pelemans, Hugo Van hamme, and Patrick Wambacq. 2017. Character-word LSTM language models. In *Proceedings of EACL*, pages 417–427.

Ivan Vulić, Nikola Mrkšić, Roi Reichart, Diarmuid Ó Séaghdha, Steve Young, and Anna Korhonen. 2017. Morph-fitting: Fine-tuning word vector spaces with simple language-specific rules. In *Proceedings of ACL*, pages 56–68.

Tian Wang and Kyunghyun Cho. 2016. Larger-context language modelling with recurrent neural network. In *Proceedings of ACL*, pages 1319–1329.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. From paraphrase database to compositional paraphrase model and back. *Transactions of the ACL*, 3:345–358.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2015. Recurrent neural network regularization. In *Proceedings of ICLR*.

Matthew D. Zeiler and Rob Fergus. 2014. Visualizing and understanding convolutional networks. In *Proceedings of ECCV*, pages 818–833.

George Kingsley Zipf. 1949. *Human behavior and the principle of least effort: An introduction to human ecology*. Martino Fine Books.