Using Morphological Knowledge in Open-Vocabulary Neural Language Models

Austin Matthews and Graham Neubig
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA, USA
{austinma,gneubig}@cs.cmu.edu

Chris Dyer
DeepMind
London, UK
cdyer@google.com

Abstract

Languages with productive morphology pose problems for language models that generate words from a fixed vocabulary. Although character-based models allow any possible word type to be generated, they are linguistically naïve: they must discover that words exist and are delimited by spaces—basic linguistic facts that are built in to the structure of word-based models. We introduce an open-vocabulary language model that incorporates more sophisticated linguistic knowledge by predicting words using a mixture of three generative processes: (1) by generating words as a sequence of characters, (2) by directly generating full word forms, and (3) by generating words as a sequence of morphemes that are combined using a hand-written morphological analyzer. Experiments on Finnish, Turkish, and Russian show that our model outperforms character sequence models and other strong baselines on intrinsic and extrinsic measures. Furthermore, we show that our model learns to exploit morphological knowledge encoded in the analyzer, and, as a byproduct, it can perform effective unsupervised morphological disambiguation.

1 Introduction

Language modelling of morphologically rich languages is particularly challenging due to the vast set of potential word forms and the sparsity with which they appear in corpora. Traditional closed vocabulary models are unable to produce word forms unseen in training data and unable to generalize sub-word patterns found in data.

The most straightforward solution is to treat language as a sequence of characters (Sutskever et al., 2011). However, models that operate at two levels—a character level and a word level—have better performance (Chung et al., 2016). Another solution is to use morphological information, which has shown benefits in non-neural models (Chahuneau et al., 2013). In this paper, we present a model that combines these approaches in a fully neural framework.

Our model incorporates explicit morphological knowledge (e.g. from a finite-state morphological analyzer/generator) into a neural language model, combining it with existing word- and character-level modelling techniques, in order to create a model capable of successfully modelling morphologically complex languages. In particular, our model achieves three desirable properties.

First, it conditions on all available (intra-sentential) context, allowing it, in principle, to capture long-range dependencies, such as that the verb agreement between “students” and “are” in the sentence “The students who studied the hardest are getting the highest grades”. While traditional n-gram based language models lack this property, RNN-based language models fulfill it.

Second, it explicitly captures morphological variation, allowing sharing of information between variants of the same word. This allows faster, smoother training as well as improved predictive generalization. For example, if the model sees the phrase “gorped the ball” in data, it is able to infer that “gorping the ball” is also likely to be valid. Similarly, the model is capable of understanding that morphological consistency within noun phrases is important. For example in Russian, one might say malen’kaya chërnïya koshka (“small black cat”, nominative), or malen’kuyu chërnïyu koshku (accusative), but malen’kiy chërnuyu koshke (mixing nominative, accusative and dative) would have much lower probability.

Third, the language model seamlessly handles out of vocabulary items and their morphological variants. For example, even if the word Obama was never seen in a Russian corpus, we expect Ya dal eto prezidentu Obame (“I gave it to presi-
dent Obama”) to have higher probability using the dative Obame than Ya dal eto prezidentu Obama, which uses the nominative. The model can also learn to decline proper nouns, including OOVs. Here it can recognize that dal (“gave”) requires a dative, and that nouns ending with “a” generally do not meet that requirement.

In order to capture these properties, our model combines two pieces: an alternative embedding module that uses sub-word information such as character and morpheme-level information, and a generation module that allows us to output words at the word, morpheme, or character-level. The embedding module allows for the model to share information between morphological variants of surface forms, and produce sensible word embeddings for tokens never seen during training. The generation model allows us to emit tokens never seen during training, either by combining a lemma and a sequence of affixes to create a novel surface form, or by directly spelling out the desired word character by character. We then demonstrate the effectiveness both intrinsically, showing reduced perplexity on several morphologically rich languages, and extrinsically on machine translation and morphological disambiguation tasks.

2 Multi-level RNNLMs

Recurrent neural network language models are composed of three parts: (a) an encoder, which turns a context word into a vector; (b) a recurrent backbone that turns a sequence of word vectors that represent the ordered sequence of context vectors into a single vector, and (c) a generator, which assigns a probability to each word that could follow the given context. RNNLMs often use the same process for (a) and (c), but there is no reason why these processes cannot be decoupled. For example, Kim et al. (2016) and Ling et al. (2015) compose character-level representations for their word encoder, but generate words using a softmax whose probabilities rely on inner products between the current context vector and type-specific word embeddings.

In our model both the word generator (§2.1) and the word encoder (§2.2) compute representations that leverage three different views of words: frequent words have their own parameters, words that can be analyzed/generated by an analyzer are represented in terms of sequences of abstract morphemes, and all words are represented as a sequence of characters.

2.1 Word generation mixture model

In typical RNNLMs the probability of the \( i \)th word in a sentence, \( w_i \) given the preceding words is computed by using an RNN to encode the context followed by a softmax:

\[
p(w_i \mid w_{<i}) = p(w_i \mid h_i) = \varphi_{\text{RNN}}(w_1, \ldots, w_{i-1}) = \text{softmax}(W h_i + b)
\]

where \( \varphi_{\text{RNN}} \) is an RNN that reads a sequence of words and returns a fixed sized vector encoding, \( W \) is a weight matrix, and \( b \) is a bias.

In this work, we will use a mixture model over \( M \) different models for generating words in place of the single softmax over words (Miyamoto and Cho, 2016; Neubig and Dyer, 2016):

\[
p(w_i \mid h_i) = \sum_{m_i=1}^{M} p(w_i \mid m_i \mid h_i)
\]

\[
= \sum_{m_i=1}^{M} p(m_i \mid h_i) p(w_i \mid h_i, m_i),
\]

where \( m_i \in [1, M] \) indicates the model used to generate word \( w_i \). To ensure tractability for training and inference, we assume that \( m_i \) is conditionally independent of all \( m_{<i} \), given the sequence of word forms \( w_{<i} \). We use three \( (M = 3) \) component models: (1) directly sampling a word from a finite vocabulary \( (m_i = \text{WORD}) \), (2) generating a word as a sequence of characters \( (m_i = \text{CHARS}) \), and (3) generating as a sequence of (abstract) morphemes which are then stitched together using a handwritten morphological transducer that maps from abstract morpheme sequences to surface forms \( (m_i = \text{MORPHS}) \). Figure 1 illustrates the model components, and we describe in more detail here:

Word generator. Select a word by directly sampling from a multinomial distribution over surface form words. Here the vocabulary is the \( |V_w| \) most common full-form words seen during training. All less frequent words are assigned zero probability by this model, and must be generated by one of the remaining models.

Character sequence generator. Generate a word as a sequence of characters. Each character is predicted conditioned on the LM hidden state \( h_i \) and the partial word generated so far, encoded
with an RNN. The product of these probabilities is the total probability assigned to a full word form.

**Morpheme sequence generator.** Similarly to the character sequence generator, we can generate a word as a sequence of morphemes. We first generate a root \( r \), followed by a sequence of affixes \( a_1, a_2, \ldots \). For example the word “devours” might be generated as devour+3\(P\)+SG+EOW. Since multiple sequences of abstract morphemes may in general give rise to a single output form, we marginalize these, i.e.,

\[
p(w_i | h_i, m_i = \text{MORPHS}) = \sum_{a_i \in \{a | \text{GEN}(a) = w_i\}} p_{\text{morphs}}(a_i | h_i),
\]

where \( \text{GEN}(a) \) gives the surface word form produced from the morpheme sequence \( a \).

Due to the model’s ability to produce output at the character level, it is able to produce any output sequence at all within the language’s alphabet. This is critical as it allows the model to generate unknown words, such as novel names or declensions thereof. Furthermore, the morphological level facilitates the model’s generation of word forms whose lemmas may be known, but whose surface form was nevertheless unattested in the training data. Finally the word-level generation model handles generating words that the model has seen many times during training.

### 2.2 Morphologically aware context vectors

Word vectors are typically learned with a single, independent vector for each word type. This independence means, for example, that the vectors for the word “apple” and the word “apples” are completely unrelated. Seeing the word “apple” gives no information at all about the word “apples”.

Ideally we would like to share information between such related words. Nevertheless, sometimes words have idiomatic usage, so we’d like not to tie them together too tightly.

We accomplish this by again using three different types of word vectors for each word in the vocabulary. The first is a standard per-type word vector. The second is the output of a character-level RNN using Long Short-Term Memory (LSTM) units (Hochreiter and Schmidhuber, 1997). The third is the output of a morphology-level LSTM over a lemma and a sequence of affixes, as output by a morphological analyzer.

Typically language models first generate a word \( w_i \) given some (initially empty) prior context \( c_{i-1} \), and then that word is combined with the context to generate a new context \( c_i \) that includes the new word. Since we have just used one or more of our three modes to generate each word, intuitively we would like to use the same mode(s) to generate the embedding used to progress the context.

Unfortunately, doing so introduces dependencies among the latent variables \( p(\text{mode} | c_i) \) in our model, making exact inference intractable. As such, we instead drop the dependency on how a word was generated and instead represent the word at all three levels, regardless of the mode(s) actually used to generate it, and combine them by concatenating the three representations. A visual representation of the embedding process is shown in Figure 2.

Additionally, should a morphological analyzer produce more than one valid analysis for a surface form, we independently produce embeddings for each candidate analysis, and combine them using a per-dimension maximum operation. Mathemati-
Figure 2: We concatenate word- morpheme- and character-level vectors to build a better input vector for our RNNLM.

-cally, the $i$th dimension of the morphological embedding $e_m$ is given by

$$e_{mi} = \max_j e_{aj}$$

where $e_{aj}$ is the embedding of the $j$th possible analysis, as computed by the LSTM over the lemma and its sequence of affixes.

The intuition behind the use of all analyses plus a pooling operation can be seen by observing the case of the word “does”, which could be $do+3$-person+singular or $doe+plural$. If this word appears after the word “he”, what we care about more is whether “does” could feasibly be a third person singular verb, thus agreeing with the subject. The max-pooling operation captures this intuition by ensuring that if a feature is active for one of these two analyses, it will also be active in the pooled representation. This procedure affords us the capability to efficiently marginalize over all three possible values of the latent variable at each step, and compute the full marginal of the word $w_i$ given the context $c_{i-1}$ during generation.

This formulation allows words with the same stem to share vector information through the character or morphological embeddings, but still affords each word the ability to capture idiomatic usages of individual words through the word embeddings. Furthermore, it allows a language model to explicitly capture morphological information, for example that third person singular subjects should co-occur with third person singular verbs. Finally, the character-level segment of the embedding allows the model to at least attempt to build sensible embeddings for completely unknown words. For example in Russian where names can decline with case this formulation allows the model to know that Obame is probably dative, even if it’s an OOV at the word level, and even if the morphological analyzer is unable to produce any valid analyses.

We combine our three-layer input vectors, our factored output model, and a standard LSTM backbone to create a morphologically-enabled RNNLM that, as we will see in the next section, performs well on morphologically complex languages.

3 Intrinsic Evaluation

We demonstrate the effectiveness of our model by experimenting on three languages: Finnish, Turkish, and Russian. For Finnish we use version 8 of the Europarl corpus, for Turkish we use the SETIMES2 corpus, and for Russian we use version 12 of the News Commentary corpus. Statistics of our experimental corpora can be found in Table 1.

Each data set was pre-processed by UNK'ing all but the top $\approx 20k$ words and lemmas by frequency. No characters or affixes were UNK'ed. This step is not strictly required—our model is, after all, capable of producing arbitrary words—but it speeds up training immensely by reducing the size of the word and lemma softmaxes. Since the morphology and/or character-level embeddings can still capture information about the original forms of these words, the degradation in modelling performance is minimal.

For morphological analysis we use Omorfi for Finnish, the analyzer of Oflazer (1994) for Turkish, and PyMorphy for Russian.

3.1 Baseline Models

Since models are not accurately comparable unless they share output vocabularies, our baselines must also allow for the generation of arbitrary word forms, including out-of-vocabulary items. We compare to three such models: an improved Kneser-Ney (Kneser and Ney, 1995) 4-gram baseline, with an additional character-level backoff model for OOVs, an RNNLM with character-level
backoff, and a pure character-based RNN language model (Sutskever et al., 2011).

Since Kneser-Ney language models (and other count-based models) are typically word-level and do not model out-of-vocabulary items, we employ a two-level approach with separate Kneser-Ney models at the word and character levels. We train the word-level model after UNK'ing low frequency words, and we train the character-level model on the same list of low frequency words. Now when we want to predict a word $w_i$ given some context $c$ we can use the word-level model to directly predict $p(w_i | c)$ unless $w_i$ is an out-of-vocabulary item. In that case we model $p(w_i | c)$ as

$$p(w_i | c) = p(\text{UNK} | c) \cdot p(w_i | \text{UNK})$$

where the first factor is the probability of the word-level model emitting \text{UNK}, and the second is the probability of the actual out-of-vocabulary word form under the character-level model.

Secondly we compare to a similar hybrid RNN model that first predicts the word-level probability for each word, and if it predicted \text{UNK} then also predicts a sequence of characters using a separate network. This model uses 256-dimensional character and word embeddings, and a 1024-dimensional recurrent hidden layer.

Finally we also compare to a standard RNN language model trained purely on the character level. For this baseline we also use 256-dimensional character embeddings and a 1024-dimensional recurrent hidden layer.

### 3.2 Multi-factored Models

For our model we use 128-dimensional word and root embeddings, 64-dimensional affix and character embeddings, 128-dimensional word-internal recurrent hidden layers for characters and morphemes, and a 256-dimensional recurrent hidden layer for the main inter-word LSTM.

The network is trained to stochastically maximize the log likelihood of the training data using Adam (Kingma and Ba, 2014). After each 10k training examples (Finnish, Turkish) or 100k training examples (Russian) we evaluate the model on a development set. If the perplexity on the development set represents a new best, we save the current model to disk, thereby mitigating overfitting via early stopping. No other regularization is used.

<table>
<thead>
<tr>
<th></th>
<th>Finnish</th>
<th>Russian</th>
<th>Turkish</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train Sents</td>
<td>2.1M</td>
<td>1.1M</td>
<td>188K</td>
</tr>
<tr>
<td>Train Words</td>
<td>38M</td>
<td>26M</td>
<td>3.9M</td>
</tr>
<tr>
<td>Dev Sents</td>
<td>1K</td>
<td>38K</td>
<td>1K</td>
</tr>
<tr>
<td>Dev Words</td>
<td>16K</td>
<td>705K</td>
<td>16K</td>
</tr>
<tr>
<td>Test Sents</td>
<td>500</td>
<td>91K</td>
<td>3K</td>
</tr>
<tr>
<td>Test Words</td>
<td>7.6K</td>
<td>1.6M</td>
<td>51K</td>
</tr>
<tr>
<td>Word Vocab</td>
<td>20K</td>
<td>21K</td>
<td>42K</td>
</tr>
<tr>
<td>Lemma Vocab</td>
<td>20K</td>
<td>20K</td>
<td>13K</td>
</tr>
<tr>
<td>Affix Vocab</td>
<td>140</td>
<td>34</td>
<td>180</td>
</tr>
<tr>
<td>Char Vocab</td>
<td>229</td>
<td>150</td>
<td>80</td>
</tr>
</tbody>
</table>

Table 1: Details of our data sets. Each cell indicates the number of sentences and the number of words in each set.

For each language we run four variants of our model. In order to preserve the ability to model and emit any word in the modelled language, it is essential that we keep the character-level part of our model intact. The morpheme- and word-level models, however, may be removed without compromising the generality of the model. As such, we present our model using only character-level input and outputs (C), using character- and morpheme-level inputs and outputs (CM), using character- and word-level inputs, but no morphol- ogy (CW), and using all three levels as per the full model (CMW).

### 3.3 Results and Analysis

Our experimental results (Table 2) show that our multi-modal model significantly outperforms all three baselines: a naive $n$-gram model, a purely character-level RNNLM, and a hybrid RNNLM for open-vocabulary language models. Furthermore, they confirm that morphological analyzers can improve performance of such language models on particularly morphologically rich languages. We observe that across all three languages the space-aware character-level model outperforms the purely character-based model that treats spaces just as any other character. Furthermore we see that the Kneser-Ney language model performs admirably well on this task, underscoring the difference in setting between the familiar, traditional closed-vocabulary LMs, and the open-vocabulary language modelling task. Additionally we find that the relative success of the $n$-gram model and the hybrid model over the character only models underscores the importance of access to word-level information, even when using a less

\[^5\]We evaluate less frequently on Russian since the dev set is much larger.
Finnish: “Russian President Vladimir Putin has an ace up his sleeve in the Ukrainian relationship.”

Table 2: Intrinsic evaluation of language models for three morphologically rich languages. Entropy for each test set is given in bits per character on the tokenized data. Lower is better, with 0.0 being perfect.

Table 3 shows some examples of sentences on which our model heavily outperforms the RNN baseline and vice-versa. We find that the sentences on which our model performs well contain much less frequent word forms. For each sentence we examine the frequency with which each token appears in our training corpus. The sentences on which our model performs best have a median token frequency of 305 times, while the sentences where the RNN performs better has an average token frequency of 3031 times. Overall our model has better log-likelihood than the RNN baseline on 88.1% of sentences.

Our methods outperform the $n$-gram model in all languages with either set of just two models, CM or CW. The same models outperform the hybrid baseline in Turkish and Russian, and achieves comparable results in Finnish. Finally, in the agglutinative languages, using all three modes performs best, while in Russian, a fusional language, characters and words alone edge out the model with morphology. We hypothesize that our morphology model is better able to model the long strings of morphemes found in Turkish and Finnish, but gains little from the more idiosyncratic fusional morphemes of Russian.

Some examples of the priors and posteriors of the modes used to generate some randomly selected sentences from the held out test set can be seen in Figure 3 and the appendix (Figure 4). The figures show that all of the models tend a priori to prefer to generate words directly when possible, but that context can certainly influence its priors. In Finnish, after seeing the word Vladmir, the model suddenly assigns significantly more weight to the following word, which is likely to be a surname. In Russian, after the preposition o, the following noun is required to be in a rare case. As such, the model suddenly assigns more probability mass to the following word being generated using the morpheme-level model.

The posteriors tell a similarly encouraging story. In Finnish we see that the word presidentillä is overwhelmingly likely to be produced by
the morphology model due to its peculiar adessive (“at”) case marking. Vladimir is common enough in the data that it can be generated wholly, but the last name Putin is again inflected into the adessive case, forming Putinnilla. Unfortunately the morphological analyzer is unfamiliar with the stem Putin, forcing the word to be generated by the character-level model. In our Turkish example, all of the short words are generated at the word level, while the primary nouns internetten (“internet”) and ders (“lecture”) are possible to generate either as words or as a sequence of morphemes. The verb, which has much more complex morphology (progressive past tense with a third person singular agent), is generated via the morphological model.

4 Extrinsic Evaluation

In addition to evaluating our model intrinsically using perplexity, we evaluate it on two downstream tasks. The first is machine translation between English and Turkish. The second is Turkish morphological analysis disambiguation.

4.1 Machine Translation

As an extrinsic evaluation we test whether our language model improves machine translation between Turkish and English. While we could transform our model into a source-conditioned translation model, we choose here to focus on testing our model as an external unconditional language model, leaving the conditional version for future work. Since neural machine translation systems struggle with low-resource languages (Koehn and Knowles, 2017), we choose to introduce the score of our LM as an additional feature to a cdec (Dyer et al., 2010) hierarchical MT system. We train on the WMT 2016 Turkish–English data set, and perform n-best reranking after re-tuning weights with the new feature.

The results, shown in Table 4 demonstrate small but significant gains in both directions, particularly into Turkish, where modelling productive morphology should be more important.

4.2 Morphological Disambiguation

Our model is a joint model over words and the latent processes giving rise to those words (i.e., which generation process was selected and, for the morpheme process, which morpheme sequence was generated). While our model is not directly trained to perform morphological disambiguation, it still performs this task quite admirably. Given a trained morphological language model, a sentence $s$, and a set of morphological analyses $z$, we can query the model to find $p(s, z) = p(w_1, w_2, \ldots, w_N)$ for a given sentence. Most notably, each $w_i$ may have a set of possible morphological analyses $\{a_1, a_2, \ldots a_{M_i}\}$ from which we would like to choose the most likely $a$ posteriori. To perform this task, we simply query the model $M_i$ times, each time hiding all but the $j$th possible analysis from the model. We can then re-normalize the resulting probabilities to find $p(a_j | s)$ for each $j \in 1 \ldots M_i$.

To make training and inference with our model tractable, we have assumed independence between previous adjacent events and the next word generation given the previous surface word forms (§2.1). Thus, the posterior probability over the analysis is only determined by the left context—subsequent decisions are independent of the process used to generate a word at time $t$. However, since disambiguating information may be present in either direction, we introduce a model variant that conditions on information in both directions. Bidirectional dependencies mean that we can no longer use the chain rule to factor the probability distribution from left-to-right. Rather we have to switch to a globally normalized, undirected model (i.e., a Markov random field) to define the probabilities of selecting the mode of generation and generation probability (conditional on the mode).

The factors used to parameterize the model are defined in terms of two LSTMs, one encoding from left-to-right the prefix of the $i$th word ($h_i$, defined exactly as above), and a second encoding from right-to-left its suffix ($h_i'$). These two vector representations are used to compute a score using a locally normalized mixture model for each word. Intuitively, the morphological analysis generated at each time step should be compatible with both the preceding words and the following words.

Optimizing this model with the same MLE criterion we used in the direct model is, unfortunately, intractable since a normalizer would need to be computed. Instead, we use a pseudo-likelihood objective (Besag, 1975).

$$
\mathcal{L}_{pl} = \prod_i p(w_i | w_{-i})
= \prod_i \sum_m p(m_i = m | w_{-i}) p(w_i | m, w_{-i})
$$

We note that although this model has a very differ-
Komünizm peşinde koşan Arnavut pek yok
Parlaklığı kaybeden mücevher: Kırılık Karadeniz’i esir alıyor
Ölülerin baskıyla karşılaştan rejim tutumu yavaşa yumuşattı. 1991 yılında çok partili → seçimleri düzenledi ve sonunda da ertesi yıl tümdeki iktidar bir kez daha batıldı.

Southeast European Times için Belgrad dan Dusan Kosanovic’in haberi - 24/06/04
23 Temmuz’dan bu yana Balkanlar la ilgili iş ve ekonomi haberlerine genel bakış:
AB’nin Genişlemeden Sorumlu Komisyon Üyesi Olli Rehn (solda) Arnavutluk Başbakanı Salı → Berişa ile 15 Mart Perşembe günü Tiran’da bir araya geldi.

Table 3: Some examples of Turkish sentence on which our morphological model heavily outperforms the baseline RNNLM (top) and some examples of the opposite (bottom). The sentences that our model performs well on have many particularly rare words, whereas the sentences the RNNLM performs well on were seen hundreds or thousands of times in the training corpus. Words in bold were seen fewer than 25 times in the training corpus. Arrows indicate line wrapping.

<table>
<thead>
<tr>
<th>Lang. Pair</th>
<th>System</th>
<th>BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>TR-EN</td>
<td>Baseline</td>
<td>15.0</td>
</tr>
<tr>
<td></td>
<td>Morph. Input</td>
<td><strong>15.2</strong></td>
</tr>
<tr>
<td>EN-TR</td>
<td>Baseline</td>
<td>10.1</td>
</tr>
<tr>
<td></td>
<td>Morph. Output</td>
<td><strong>10.5</strong></td>
</tr>
</tbody>
</table>

Table 4: Machine Translation Results

Ent semantics from the directed one, the PL training objective is identical to the directed model’s, the only difference is that features are based both on the past and future, rather than only the past.

Similarly to training, evaluating sentence likelihoods using this model is intractable, but posterior inference over $m_i$ and $a_i$ is feasible since the normalization factors cancel and therefore do not need to be computed.

For our experiments we use the data set of Yuret and Türe (2006) who manually disambiguated from among the possible forms identified by an FST. We significantly out-perform the simple baseline of randomly guessing, and our results are competitive with Yatbaz and Yuret (2009), although they evaluated on a different dataset so they are not directly comparable. Furthermore, we also compare to a supervised model (Shen et al., 2016). While unsupervised techniques can’t hope to exceed supervised accuracies, this comparison provides insight into the difficulty of the problem. See Table 5 for results.

5 Related Work

Purely Character-based or Subword-based LMs have a rich history going all the way back to Markov (1906)’s work modelling Russian character-by-character with his namesake models. More recently Sutskever et al. (2011) were the first to apply RNNs to character-level language modelling, leveraging their ability to handle the long-range dependencies required to model language at the character level. It is also possible to alleviate the closed vocabulary problem by training models on automatically acquired subword units (Mikolov et al., 2012; Sennrich et al., 2015). While these approaches allow for an open vocabulary (or nearly open, in the case of subwords) they discard a large amount of higher-level information, inhibiting learning.

Character-aware language models, which combine character- and word-level information have shown promise (Kang et al., 2011; Ling et al., 2015; Kim et al., 2016). Unsupervised morphology has also been shown to improve the representations used by a log-bilinear LM (Botha and Blunsom, 2014). Jozefowicz et al. (2016) explore many interesting such architectures, and compare with fully character-based models. While these models allow for the elegant encoding of novel word forms they lack an open vocabulary.

Open-vocabulary hybrid models alleviate this problem, extending the benefits of character-level representations to the generation. Such hybrid models with open vocabularies have been around since Brown et al. (1992). More recently, Chung et al. (2016) and Hwang and Sung (2016) describe methods of modelling sentences at both the word and character levels, using mechanisms to allow both a word-internal model that captures short-range dependencies and a word-external model to capture longer-range dependencies. These models have been successfully applied to machine
translation by Luong and Manning (2016), who use a character-level model to predict translations of out of vocabulary words. Our work falls in this category—we combine multiple representation levels while maintaining the ability to generate any character sequence. In contrast to these previous works, we demonstrate the utility of incorporating morphological information in these open-vocabulary models.

**Mixture model language generation** where the mixture coefficients are predicted by a neural net are becoming quite common. Neubig and Dyer (2016) use this strategy to combine a count-based model and a neural language model. Ling et al. (2016) interpolate between character- and word-based models to translate between natural language text and computer code. Merity et al. (2016) also use multiple output models, allowing a word to either be generated by a standard softmax or by copying a word from earlier in the input sentence.

## 6 Conclusion

We have demonstrated a technique for language modelling that works particularly well on morphologically rich languages where having an open vocabulary is desirable. We achieve this by using a multi-modal architecture that allows words to be input and output at the word, morpheme, or character levels. We show that knowledge of the existence of word boundaries is of critical importance for language modelling tasks, even when otherwise operating entirely at the character level, resulting in a surprisingly large reduction in per-character entropy across all languages studied.

Furthermore, we demonstrate that if we have access to a morphological analyzer we can leverage it to further improve our LM, reinforcing the notion that the explicit inclusion of linguistic information can indeed aid learning of neural models.

## Acknowledgements

We would like to thank Sebastian Mielke for his insightful discussion and feedback on this work. This work is sponsored by Defense Advanced Research Projects Agency Information Innovation Office (I2O). Program: Low Resource Languages for Emergent Incidents (LORELEI). Issued by DARPA/I2O under Contract No. HR0011-15-C0114. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

## References


<table>
<thead>
<tr>
<th>Model</th>
<th>Supervised?</th>
<th>Ambiguous Words</th>
<th>All words</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random Chance</td>
<td>no</td>
<td>34.08%</td>
<td>52.66%</td>
</tr>
<tr>
<td>Unidirectional</td>
<td>no</td>
<td>55.15%</td>
<td>80.28%</td>
</tr>
<tr>
<td>Bidirectional</td>
<td>no</td>
<td><strong>63.85%</strong></td>
<td><strong>84.11%</strong></td>
</tr>
<tr>
<td>Shen et. al</td>
<td>yes</td>
<td>91.03%</td>
<td>96.43%</td>
</tr>
</tbody>
</table>

Table 5: Morphological disambiguation accuracy results for Turkish.


A  Example sentences

Finnish: “Russian President Vladimir Putin has an ace up his sleeve in the Ukrainian relationship.”

Turkish: “He gave only two lectures on the internet this year.”

Russian: “The investigation does not theorize about the attacker’s motives.”

Figure 4: Some examples of the priors (left) and posteriors (right) over modes used to generate each word in some sample sentences. Probability given to the word-, morpheme-, and character-level models are shown in red, blue, and gold respectively. The Finnish example is a reprint of Figure 3.